CAES

GR712RC memory production test coverage and usage constraints

Technical note 2022-10-11

Doc. No GR712RC-TN-0002

Issue 1.1

CHANGE RECORD

Issue	Date	Section / Page	Description
1.0	2022-03-18	All	First issue of this document
1.1	2022-10-11	4, 5.2 (new section), 5.3, 6.2	Minor changes in wording. (4) Clarified that FTAHBRAM issue occurs with "back-to-back" in addition to "burst" writes. (5.2) New section on how the production test was validated. (5.3) Clarified the definition of "address" and "bit-state" coverage. (6.2) Added a statement about TID.

Table of Contents

1 INTRODUCTION

1.1 Scope of the document

The GR712RC is a microprocessor implementing two independent CPUs on one silicon die, i.e. CPU0 and CPU1. For each CPU, the silicon implementation utilizes memory macrocells to implement various functions like the register file, and the cache memories. Memory macrocells are also used for implementing the FTAHBRAM (On-chip Memory with EDAC Protection) on the silicon, as well as the instruction trace buffers and the AHB trace buffer. The trace buffers are used for debug purposes only. For the remainder of this document, the memory macrocells on the silicon will also be referred to as simply *memories*.

During the production of digital parts, specific production tests are deployed to verify the functionality of such memory macrocells and to screen out any defective parts as part of the overall production test program. The production tests specific for the memory macrocells of the GR712RC part are implemented by executing software on the two CPUs since Built-In-Self-Test (BIST) has not been implemented for the GR712RC.

In the beginning of 2022, it was discovered that revision 3 of the GR712RC production test program did not offer full coverage of all storage bits in the memory macrocells. Consequently, the production test program was updated by adding additional subtests related to the memory macrocells resulting in revision 5 of the production test program. This Technical Note covers three partially separate issues discovered during the update process:

1. CPU1 coverage issue: One of the subtests of revision 3 of the production test program was only executed on CPU0, hence the test coverage of the memory macrocells specific to CPU1 was incomplete.
2. CPU0 coverage issue: Although revision 3 of the production test program tested all word locations and storage bits in the memory macrocells specific to CPU0, not all storage bits were tested in both logical states.
3. FTAHBRAM back-to-back write issue: Data errors can be induced into the FTAHBRAM during back-to-back writes at low supply voltage.

This Technical Note details the precise test coverage of the memory macrocells in production test program revision 3; the observed and extrapolated probabilities for GR712RC parts with defects to have evaded screening; the consequences for software running on a GR712RC part with defects in the memory macrocells; a description of the FTAHBRAM back-to-back write issue; and a software-based method for testing for CPU0 and CPU1 related defects in parts already integrated into boards.

1.2 Parts screened with incomplete test coverage of memory macrocells

The CPU1 and CPU0 test coverage issues apply to GR712RC parts screened with revision 3 and earlier of the production test program. In particular, all delivered GR712RC parts with date codes before D2144 were screened with revision 3 of the production test program. Parts screened with revision 5 and later of the production test program have full test coverage of all memory macrocells.

The FTAHBRAM back-to-back write issue applies to all GR712RC parts regardless of date code and what revision of the production test program it was screened with.

The following products have complete coverage of memory macrocells:

- GR740 – uses BIST for memory testing
- GR716 – uses BIST for memory testing
- GR718B – does not use memory macrocells

1.3 Probability that a delivered part has a memory defect

Parts tested with revision 5 of the production test program are guaranteed to be fully tested for defects in memory macrocells. Parts screened with revision 3 of the production test program may have defects not detected by the screening. Estimated probabilities are given below.

CPU1: In the first batch tested after revision 5 of the production test program was introduced, 1.2% of tested parts have been observed to have defects in CPU1 memories that would not have been discovered in revision 3 of the production test program.

CPU0: No parts tested with revision 5 of the production test program have been found to have defects in CPU0 memories that would not have been discovered using revision 3 of the production test program. The probability that a given part screened with revision 3 of the production test program has a defect in a CPU0 memory that could cause data corruption or a software crash has been extrapolated to be less than 0.004%. Existence of such a defect is expected to be discovered immediately when running non-trivial software on the part.

FTAHBRAM back-to-back writes: Neither revision 3 nor revision 5 of the production test program includes any screening of the FTAHBRAM back-to-back write issue. Errors may be induced during back-to-back writes to the FTAHBRAM in any GR712RC part. The exact characteristics vary from part to part. Refer to the errata list in the GR712RC user’s manual [RD2] for workarounds.

1.4 Testing for memory defects in parts already integrated into boards

For customers with GR712RC parts tested with revision 3 of the production test program already integrated into boards, the software package “gr712rc-tn0002-sw” is provided. The software package contains source code and a prebuilt binary for board-level tests of all CPU0 and CPU1 memories. See section 7 for details.

Software is not provided to test for the FTAHBRAM back-to-back write issue.

1.5 Distribution

Contact Cobham Gaisler for inquiries about redistribution of this Technical Note.

1.6 Contact

For questions on this document, please contact Cobham Gaisler support at support@gaisler.com.

1.7 Reference documents

[RD1] "Radiation characterization of a dual core LEON3-FT processor", F. Sturesson, J. Gaisler, R. Ginosar, T. Liran, 2011, 12th European Conference on Radiation and Its Effects on Components and Systems, DOI: 10.1109/RADECS.2011.6131334

[RD2] "GR712RC – Dual-Core LEON3FT SPARC V8 Processor – User's manual", Cobham Gaisler, GR712RC-UM, Available at https://www.gaisler.com/gr712rc

[RD3] "Handling of External Memory EDAC Errors in LEON/GRLIB Systems", Cobham Gaisler, GRLIB-AN-0004, Available at https://www.gaisler.com/notes

[RD4] "GR712RC – Dual-Core LEON3-FT SPARC V8 Processor – Data Sheet", Cobham Gaisler, GR712RC-DS. Available at https://www.gaisler.com/gr712rc

1.8 Abbreviations

ATE	Automatic Test Equipment
BIST	Built-In Self-Test
CPU	Central Processing Unit
DUT	Device Under Test
ECC	Error Correcting Code
EDAC	Error Detection and Correction
FPU	Floating Point Unit
GEO	Geostationary Earth Orbit
IU	Integer Unit
MMU	Memory Management Unit
RTL	Register Transfer Level
SEU	Single-Event Upset
SMP	Symmetric Multi-Processing
TID	Total Ionizing Dose
TLB	Translation Look-aside Buffer

2 EXECUTIVE SUMMARY

All GR712RC parts are tested at several stages of the production to screen out parts with defects. However, it has been discovered that revision 3 of the GR712RC production test program had incomplete coverage of some of the memory macrocells. In response, the production test program was updated and revalidated. Revision 5 of the production test program has full coverage of all the CPU memories. However, parts that passed revision 3 of the production test program may contain defects in the partially tested CPU memories.

While revising and revalidating the production test program, three issues were identified in revision 3 of the production test program:

1. **CPU1 coverage issue**
   a. The CPU1 caches and IU register file were **not** tested with full word and bit coverage.
   b. **77%** of all data words and **81%** of all **bit-states** were **not** tested. See section 5.3 for the definition of "bit-states" and additional information.
   c. Untested bits existed in all CPU1 memories, but most were located in the D-cache.
2. **CPU0 coverage issue**
   a. The CPU0 caches and IU register file were tested with **full coverage** of addresses and bits.
   b. However, **4%** of all **bit-states** were not tested.
   c. The partially tested bits were located in the cache tag memories (instruction, data, snoop), and in the IU register file.
3. **FTAHBRAM back-to-back writes**
   a. The 192 KiB FTAHBRAM is tested with **full coverage** of addresses and bit-states using **single word writes**.
   b. However, data errors may be induced during **back-to-back writes** to the FTAHBRAM.
   c. The rate of errors increases with decreasing supply voltage and there is a large part to part variation. See section 4.
   d. Single word writes to the FTAHBRAM do not induce errors. Refer to the errata list in the GR712RC user's manual [RD2] for workarounds.

After the introduction of revision 5 of the production test program, 1.2% of tested parts have been found to have defects in CPU1 memories that would not have been discovered in revision 3 of the production test program.

No parts tested with revision 5 of the production test program have been found to have defects in CPU0 memories that would not have been discovered in revision 3 of the production test program. The probability of occurrence is estimated in section 5.4.

The consequences for software encountering a defect vary depending on the type and location of the defect. For example, a single stuck bit in the IU register file would result in the CPU becoming unresponsive due to an infinite error-correction loop. In contrast, a single stuck bit in a cache memory appears to software as a cache miss, and software execution continues without error. See section 6 for further failure mechanisms and section 5.4 for estimated probabilities of defects in each memory type.

Parts already integrated into boards can be tested for occurrence of defects in CPU0 and CPU1 memories with a software package provided by Cobham Gaisler (see section 7). Software is not provided to test for the FTAHBRAM back-to-back write issue.

3 INVENTORY OF GR712RC MEMORY MACROCELLS

A graphical overview of memory macrocells in the GR712RC is given in Figure 1. The dashed rectangle indicates memories with incomplete *address coverage* in production test program revision 3. The red dots indicate memories with incomplete *bit-state coverage* (see section 5.3 for definitions). Other memories, such as the FPU registers, the 16-entry MMU TLB and FIFOs are not shown since they are implemented as hardened flip-flops [RD1], not memory macrocells. In the remainder of this Technical Note, the term “memories” will mean “memory macrocells”.

A size comparison of the memories specific to one CPU is given in Figure 2 (CPU0 and CPU1 are identical in this regard).

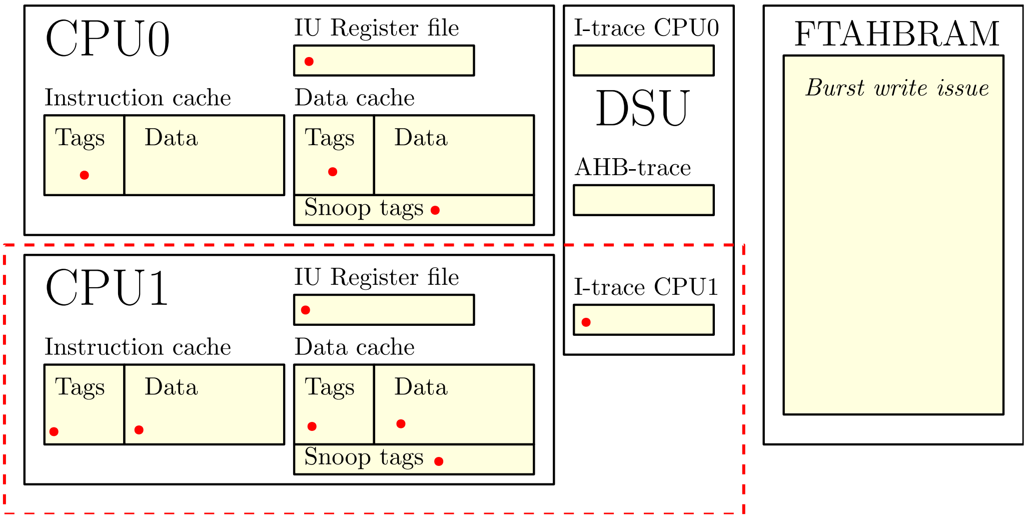


Figure 1 Overview of memory macrocells in the GR712RC. Red dots: Not all bit-states tested. Dashed rectangle: Address decoding not fully tested. FTAHBRAM: Has burst/back-to-back write issue.

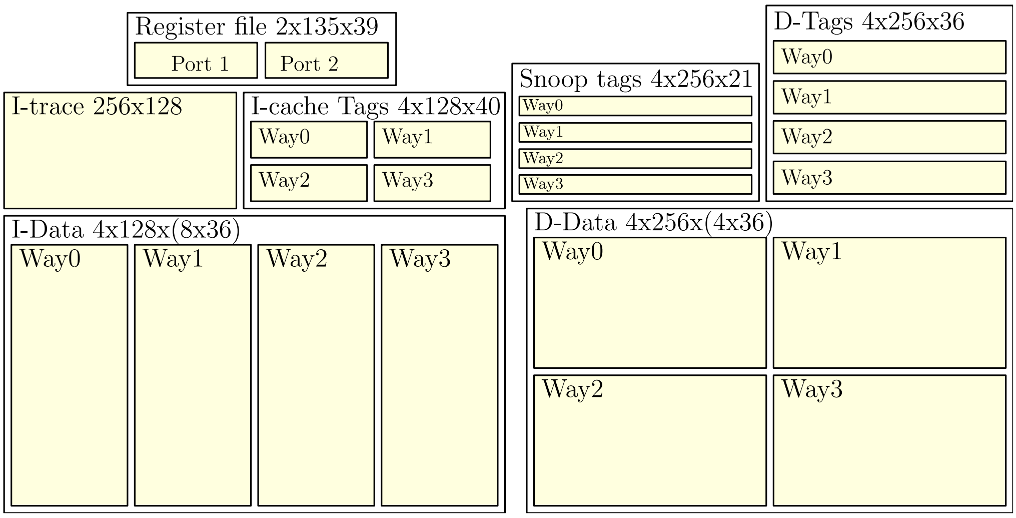


Figure 2 Comparison of the relative sizes of memories specific to one CPU (either CPU0 or CPU1). The area of each shaded rectangle is proportional to the number of bits it contains.

The organization and error detection mechanisms of the memories is summarised in the text below. The percentages do not include the instruction trace buffer since this buffer should not be used by application software. Additional details about tag fields can be found in the GR712RC user’s manual

[RD2] sections 4.3.2, 4.4.3, 4.6.5, and 4.8.4.

1. IU register file (2x135x39 bits, 2.7% of CPU memory bits)
    a. Contains the %g0-7, %i0-7, %l0-7 and %o0-7 registers.
    b. Each register is 32 (data) + 7 (ECC) = 39 bits wide.
    c. 1-bit error correction, 2-bit error detection (BCH code)
    d. There are 8 global registers + 8 windows of 16 registers = 136 logical registers
    e. The global register %g0 is 0 and not stored in memory => 135 registers in memory
    f. The IU register file has two read ports and this is implemented by storing all registers in two identical copies in two memories. => 2x135x39 bits total size
2. I-cache tags (4x128x40 bits, 5.3% of CPU memory bits)
    a. 4 cache ways
    b. Each way has 128 cache lines
    c. Each cache line tag contains the virtual address (20 bits), MMU context (8 bits), valid bits (8 bits), and parity bits (4 bits) for a total of 40 bits.
    d. 1-bit error detection (parity)
3. I-cache data (4x128x8x36 bits, 38.4% of CPU memory bits)
    a. There are 4 cache ways
    b. Each way has 128 cache lines
    c. Each line contains 8 words
    d. Each word has 32 (data) + 4 (parity) bits
    e. 1-bit error detection (parity)
4. D-cache tags (4x256x36 bits, 9.6% of CPU memory bits)
    a. Differs from I-cache tags only in that there are 4 valid-bits instead of 8 and twice as many lines (256).
5. D-cache data (4x256x4x36 bits, 38.4% of CPU memory bits)
    a. Differs from I-cache data only in that there are twice as many lines (256) with each line being half the size (4 words).
6. Snoop tags (4x256x21 bits, 5.6% of CPU memory bits)
    a. Each snoop tag has 20 address bits (the physical address of a D-cache line) and 1 parity bit.
    b. 1-bit error detection (parity)
7. Instruction trace-buffer (256x128 bits). This memory is only used for debug purposes, not in applications.
8. AHB trace buffer (256x128 bits). There is a single copy of this memory shared by all AHB masters and slaves. This memory is only used for debug purposes, not in applications.
9. FTAHBRAM (48x1024x40 bits). There is a single copy of this memory. It is a general-purpose storage area for CPUs and DMA peripherals.

The precise coverage of revision 3 of the production test program of these memories is provided in section 5. Consequences for software executing on a GR712RC part with at least one defect in a CPU memory are given in section 6.

In short, integrity of the IU register file (1) is critical in all operation modes except power-down. Integrity of cache tags and data (2-5) is critical in any application where caches are enabled. Integrity of snoop tags (6) is critical to maintain cache coherency in SMP applications as well as single-core applications that make use of DMA. The integrity of trace buffers (7) and (8) are inconsequential to application software. Finally, from a software perspective, errors in (9) are equivalent to errors in external memory, and application note [RD3] applies.

4 FTAHBRAM BACK-TO-BACK WRITE ERRORS

The FTAHBRAM is fully tested in production test program revision 3 using single-word accesses. However, it has been found that data errors can be induced when the memory is written using back-to-back writes:

- Single-bit errors can be induced in the first word of the back-to-back write sequence.
- All errors that have been observed would have been corrected transparently on readout by the built-in EDAC of the FTAHBRAM.
- Errors have been observed only in limited address ranges.
- A large part to part variability has been observed.
- For a given part, the number of memory locations where errors may be induced increases with decreasing supply voltage.
- In some parts errors can be induced at nominal (1.8 V) supply voltage.
- Neither single-word writes, nor back-to-back reads induce data errors.

Back-to-back writes are generated in three circumstances in the GR712RC:

1. When a CPU executes doubleword store instructions "`std`" and "`stda`".
2. By cores implementing DMA. For example, the GRSPW2 (SpaceWire Interface with RMAP Support), GRETH (Ethernet Media Access Controller (MAC)), and B1553BRM (MIL-STD-1553B BC/RT/BM) cores.
3. When two AHB masters (for example CPU0 and CPU1, or CPU0 and a DMA core) attempt to write to the same AHB slave during the same or adjacent clock cycles.

Refer to the errata 1.7.21 of the GR712RC user's manual [RD2] for workarounds.

5 COVERAGE OF PRODUCTION TEST PROGRAM REVISION 3

A simplified overview of the GR712RC production test program is shown in Figure 3. Each rectangle indicates a subset of tests and is grouped according to whether it tests CPU0, CPU1 and/or the FTAHBRAM.

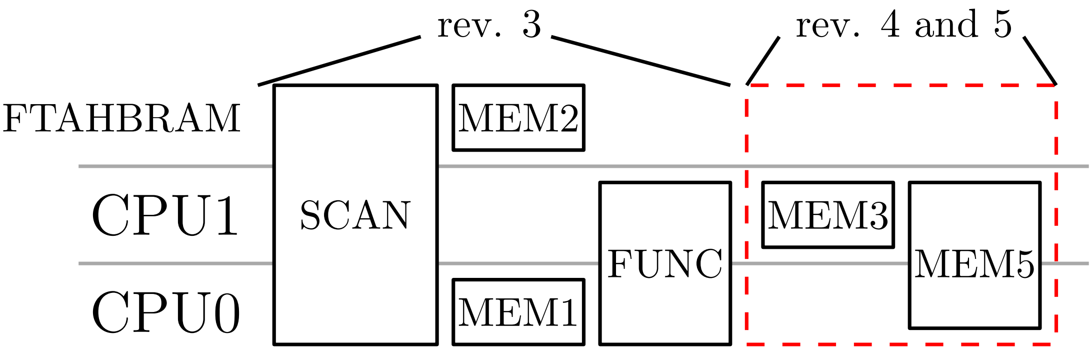


Figure 3 Simplified overview of subtests in production test program revision 3-5.

A short description of each subtest from Figure 3 follows below:

1. SCAN: The GR712RC is placed in a proprietary scan-mode that verifies functionality of the internal flip-flops. Does not test memories (i.e. memory macrocells).
2. MEM1: At-speed (100 MHz) software-based test of CPU0 memories and AHB trace buffer. Covers all word addresses and bits, but there are bits in the IU register file, the I- and D-cache tag memories and the D-cache snoop tag memory that are not tested in both logical states. See section 5.3 for details.
3. MEM2: At-speed (100 MHz) software-based test of FTAHBRAM. Fully tests all words and bit-states, but only using single word writes and reads (no back-to-back accesses).
4. FUNC: At-speed (100 MHz) software-based functional tests. Implicitly tests a significant fraction of the IU register file and instruction caches of both CPUs. See section 5.3 for details.
5. MEM3: A variant of the MEM1 subtest that is executed on CPU1 instead of CPU0. The MEM3 subtest has the same coverage problems as the MEM1 subtest. This subtest was added in revision 4 of the production test program.
6. MEM5: Tests of IU register file and cache tag memories that cover the gaps in the MEM1 and the MEM3 subtests. The MEM5 subtest was added in revision 5 of the production test program.

All tests are repeated at the corners of the recommended operating supply voltage and temperature specified in the GR712RC datasheet [RD4].

5.1 Statistics on number of parts with defects

The probability that a part that passed revision 3 of the test program has at least one undetected defect in internal memories can be directly estimated using data from the first batch tested after the introduction of the MEM3 and MEM5 subtests. Out of the parts in this batch that passed the revision 3 subtests (SCAN, MEM1, MEM2, and FUNC), 1.2% failed the MEM3 subtest. Furthermore, all parts in this batch that passed the MEM1 and MEM3 subtests also passed the MEM5 subtest. Therefore, the estimated probability of a residual defect being present in CPU1 memories after revision 3 screening is 1.2%. There is insufficient statistics to estimate corresponding probability for CPU0 memories.

The two CPUs in the GR712RC are essentially identical. Hence the fraction of parts that fail only in the MEM1 subtest (and pass all other subtests) in revision 3 of the production test program is a good proxy indicator for the fraction of parts that would have failed only in the MEM3 subtest. Inspection of records from thousands of parts tested with revision 3 of the production test program showed that about 1% of the parts failed only in the MEM1 subtest. This is consistent with the observed 1.2% rate of “unique” MEM3 fails above.

All parts that have failed only in the MEM1 or MEM3 subtests have done so at all tested supply voltage and temperature corners. There is no observed voltage or temperature dependence.

The production test records do not contain enough information to distinguish between different types of defects for a given failing part. For that reason, no attempt has been made to quantify the rate of address decoding errors relative to defects localized to single bits (e.g. stuck bits, see Figure 7).

5.2 Production test validation architecture

The architecture of the physical production tests is shown in Figure 4. In the case of the MEM1-5 subtests, software is executed on the device under test (DUT) by means of an ATE that emulates external memory. The software executes a memory test pattern by making diagnostic reads and writes and reports the results back to the ATE.

For full insight into the internals of the GR712RC, the validation of the production test program makes use of an RTL simulation (illustrated in Figure 5) that includes simulation models of both the GR712RC and an ATE. Also shown in Figure 5 are three instrumentation points, each allowing an independent method of validation and coverage measure:

1. Instruction trace (a simplified excerpt is shown).
2. Memory access trace (a simplified excerpt is shown).
3. Stuck-at fault injection (an excerpt from a TCL script handling injection is shown).

The memory access trace (2) provides the most direct view. It shows all read and write accesses to all internal memories. However, relying only on this trace for verification would result in false positives. Due to the processor microarchitecture, some memory locations may be read, but discarded with the contents having no effect on program execution. One example where this happens is during diagnostic readout of a memory with ECC. For each memory location, two diagnostic accesses are needed: the first for reading the data bits and the second for reading ECC bits. But it is not possible to tell from the memory access trace alone whether the data or ECC bits are being accessed.

The instruction trace (1) complements the memory access trace. On this level of abstraction, it is possible to see exactly which type of diagnostic access is made, where it is made to, and which part of the data is used. This makes it possible to directly compare software execution against the test specification (see section 5.3). But doing so requires some modelling of the CPU internals when parsing the instruction trace. For example, keeping track of register contents and mapping diagnostic accesses to memory primitives. An example of the latter is the 3-port IU register file. In the GR712RC the three ports are implemented using two physical copies with a common write port, but separate read ports (see Figure 6). Both copies must be tested for defects[1]. The complexity necessitates the validation also of the software that performs the parsing. Such validation has been performed by interleaving the instruction and memory traces to confirm that all memory accesses inferred by the instruction parsing software actually take place.

The third method is the most powerful. Instrumentation of the memory simulation models allows simulated defects to be introduced prior to the execution of the test software. When the full simulated test flow is executed, it is verified that the stuck bit results in a behavioral change that would have been detected by the ATE. Any error successfully detected in such a simulation would also have been detected in a real production test. The major disadvantage of this method is that the simulation is too time consuming to cover all 384290 bits/CPU (see section 3) in a reasonable amount of time. The coverage of the CPU0 and CPU1 IU register files (10530 bits/CPU) of production test program revision 5 has been fully tested with this method. The coverage of remaining memories has instead been verified using error injection on randomly chosen bits. Picking error injection locations uniformly at random removes observer bias and allows stringent statistical lower bounds on coverage to be set with a relatively small number of simulations.

[1] This aspect was handled correctly in production test program revision 3.

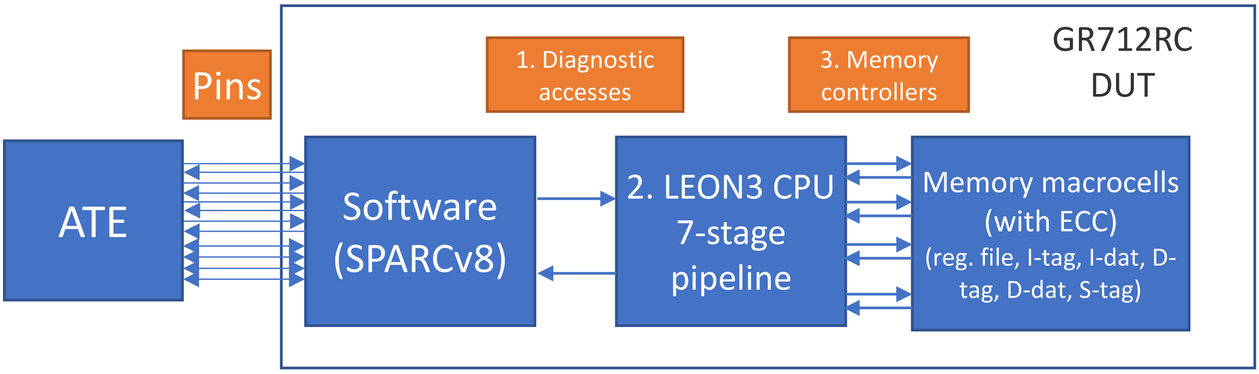
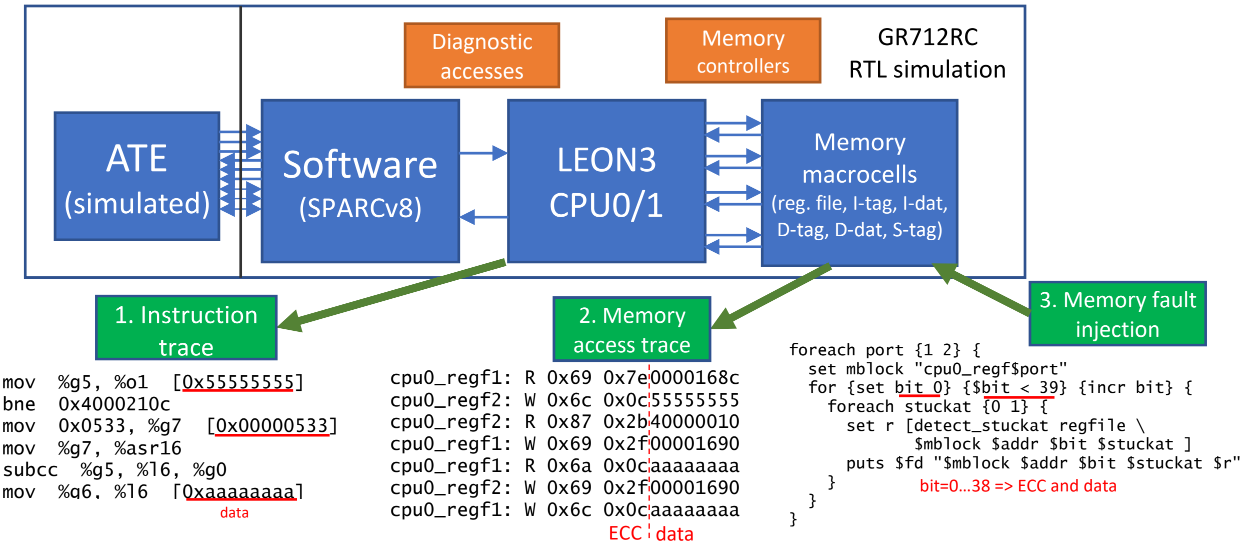

Pins
ATE
1. Diagnostic accesses
3. Memory controllers
GR712RC DUT
Software (SPARCv8)
2. LEON3 CPU 7-stage pipeline
Memory macrocells (with ECC)
(reg. file, I-tag, I-dat, D-tag, D-dat, S-tag)

Figure 4 *Memory test architecture in production test program (applies to both revision 3 and 5).*



```
mov  %g5, %o1  [0x55555555]
bne  0x4000210c
mov  0x0533, %g7  [0x00000533]
mov  %g7, %asr16
subcc  %g5, %l6, %g0
mov  %g6, %l6  [0xaaaaaaaa]
```

```
cpu0_regf1: R 0x69 0x7e0000168c
cpu0_regf2: W 0x6c 0x0c55555555
cpu0_regf2: R 0x87 0x2b40000010
cpu0_regf1: W 0x69 0x2f00001690
cpu0_regf1: R 0x6a 0x0caaaaaaaa
cpu0_regf2: W 0x69 0x2f00001690
cpu0_regf1: W 0x6c 0x0caaaaaaaa
```

```
foreach port {1 2} {
  set mblock "cpu0_regf$port"
  for {set bit 0} {$bit < 39} {incr bit} {
    foreach stuckat {0 1} {
      set r [detect_stuckat regfile \
                 $mblock $addr $bit $stuckat ]
      puts $fd "$mblock $addr $bit $stuckat $r"
    }
  }
}
```

Figure 5 *Architecture of environment for validation and coverage computation of new and old production test programs.*

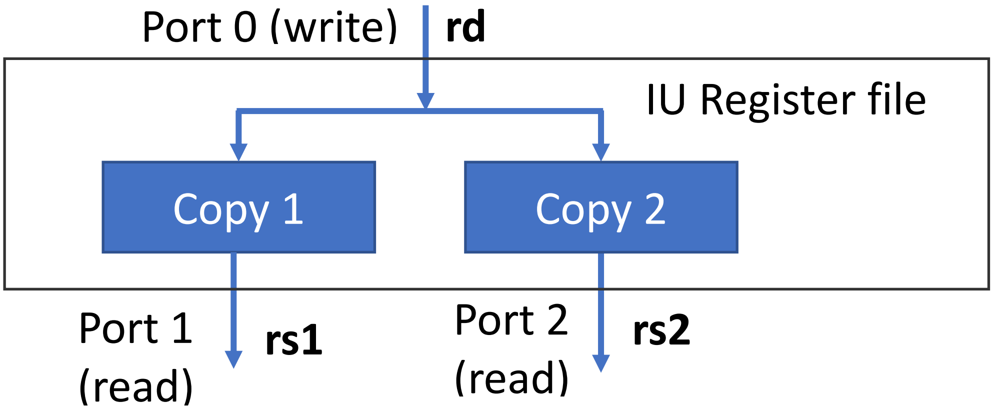


Figure 6 *Implementation of 3-port IU register file in the GR712RC. A SPARC instruction references up to three registers and each corresponds to a particular port of the register file. Specifically, destination register ("rd"), and source registers 1 and 2 ("rs1" and "rs2").*

5.3 Memory test coverage of production test program revision 3

As indicated by Figure 3, the main coverage issue in production test program revision 3 is the lack of dedicated tests for CPU1 memories; the only subtest with partial coverage of CPU1 memories is FUNC. A second issue is the coverage gap between the MEM1 and MEM5 subtests for CPU0 memories. This section quantifies the incomplete coverage of the FUNC subtest for CPU1 memories and the MEM1 subtest for CPU0 memories.

In terms of coverage, a particular bit cell is considered fully tested if the following three conditions hold:

1. The production test program writes 0 to the bit cell and later reads it back as 0.
2. The production test program writes 1 to the bit cell and later reads it back as 1.
3. The access patterns (for example, address sequence, checkerboard words, read/write timing) recommended for memory testing by the ASIC technology supplier were followed when writing and reading the bit cell. There are separate patterns for “address” and “bit-state” coverage. See section 7.1 for details.

According to the above definition, revision 5 of the production test program has full coverage of all bit cells in both CPUs (determined using the framework described in section 5.2). On the other hand, revision 3 of the production test program has incomplete coverage of both CPU0 and CPU1. However, there are multiple possible measures of coverage.

One idealized measure is the following: Given a device with a memory defect, the coverage measure could be defined as the probability that the production test program would detect the defect. This measure would allow the computation of the probability that a part has a defect after it has been screened with production test program revision 3. The disadvantage of this measure is that it is unwieldy to compute since the precise probability depends on the relative rates of different failure mode and these rates are poorly constrained by available statistics (see section 5.1). Instead, we use two simpler measures of coverage as approximations.

To obtain quantitative measures of coverage, two classes of memory failure modes need to be considered. Defects can be broadly categorized in two types

1. Localized: Defects localized to a single bit-cell (and possibly its immediate neighbors).
2. Distributed: For example, defects in memory control signals such as address or data lines.

The difference is illustrated in Figure 7.

A localized defect generally has the effect that a single bit is read as 1 when it should be 0 or vice versa under some circumstance. To for the presence of such a defect in a particular bit requires at least that the bit is read as both 0 and 1 at some points during the test. Therefore the number of “bit-states” that a particular test software reads provides a measure of the test’s coverage. We call this the “bit-state coverage”, it is defined as follows:

- Suppose that a memory is composed of N physical bits ($2N$ “bit states”).
- The production test writes and reads n_0 of them in the 0-state.
- The production test writes and reads n_1 of them in the 1-state.
- Then the reported coverage percentage is $(n_0 + n_1)/(2N)$.

This simplified measure corresponds to the probability of defect detection if the following assumptions hold:

- All defects are localized.
- All defects are stuck-at defects with equal probability of stuck-at-0 and stuck-at-1.
- All bit cells are *a priori* equally likely to be affected by a defect.

This is the coverage measure reported in Table 2.

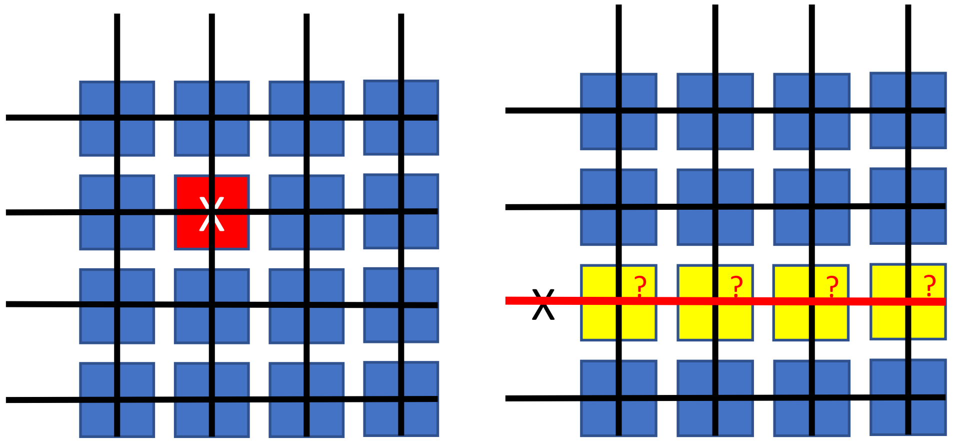


Figure 7 *Illustration of two classes of defects. Left: A single memory cell (bit) is defective. Functional errors can only be observed when addressing the defective bit. Right: One of the data, address or control signals is defective. Functional errors may be observed when accessing any bit impacted by the distributed defect.*

A distributed defect typically causes writes and reads to change or be affected by data at a different memory location than the one being addressed. In this case it is less clear how to quantify the coverage of a test with incomplete coverage. A method to test for such errors is to first write unique data to each memory address and then read back the entire memory. Therefore, we define the "address coverage" of a test the fraction of memory words written and read during the test. More precisely:

- Suppose that a memory is composed of N physical "words" (each with a separate address).
- The production test writes non-zero values into n distinct words during its execution.
- Then the coverage percentage is defined to be n/N.

This is the measure reported in Table 1. It is not directly related to any probability.

The coverage measures above were determined for revision 3 of the production test program by running the MEM1 and FUNC subtests in the validation framework shown in Figure 5. The coverage percentages were obtained using the instruction trace method for CPU0/MEM1 and the memory trace method for CPU1/FUNC. The error injection feature was used to validate selected results. I.e., that the production test program really does detect defects in bits that were determined to be covered and did not detect defects in uncovered bits.

Snoop tag coverage was analysed for CPU0/MEM1, but not CPU1/FUNC. In the latter case a lower bound of zero coverage has been assumed. This is justified because only a small fraction of snoop tag defects can potentially be discovered by the FUNC subtest. Firstly, because CPUs do not snoop on themselves. Secondly because during execution of the FUNC subtest only a small amount of memory is cached in both the CPU0 and CPU1 D-caches (5 cache lines).

The address coverage estimate is given in Table 1 and the bit-state coverage estimate in Table 2. Note that the estimated bit-state coverage is higher than address coverage for cache tag memories. This happens because words containing all 0s have been excluded from the analysis behind Table 1, but not from Table 2. The coverage by the FUNC subtest is incomplete for all CPU1 memories, both in terms of addresses and bit-states. D-cache coverage is negligible, while the IU register file and I-cache have partial coverage. Therefore, residual errors in CPU1 memories after the FUNC subtest screening is expected to be somewhat more common in the D-cache than in other memories. For CPU1 memories, nothing can be inferred about the rate of address decoding errors in comparison with bit errors.

The MEM1 subtest has 100% address coverage of CPU0 memories. Not only according to the simplified measure, but also according to the more stringent conditions in the beginning of this section. Therefore, any residual defects in CPU0 memories after MEM1 screening can be assumed to be localized to single bits. The bit-state coverage of the MEM1 subtest is incomplete in the IU register file and tag memories with 4% of bit-states not tested (because 8% of all bits are only tested in a single logical state). However, note that these numbers neglect the FUNC coverage of CPU0 memories.

Table 1 "Address coverage" of CPU0 memories by the MEM1 subtest, and CPU1 memories by the FUNC subtest.

	CPU0 (MEM1)		CPU1 (FUNC)	
	Words	Fraction	Words	Fraction
IU register file	270	100%	160	59%
I-cache tags	512	100%	208	41%
I-cache data	4096	100%	1242	30%
D-cache tags	1024	100%	71	6.9%
D-cache data	4096	100%	74	1.8%
Snoop tags	1024	100%	0	0.0%
Totals:	11022	100%	1755	16%

Table 2 "Bit-state coverage" of CPU0 memories by the MEM1 subtest, and CPU1 memories by the FUNC subtest. Note that the total number of bit-states is twice the total number of memory bits.

	CPU0 (MEM1)		CPU1 (FUNC)	
	Bit-states	Fraction	Bit-states	Fraction
IU register file	18444	88%	8937	42%
I-cache tags	33100	81%	21856	53%
I-cache data	294912	100%	44870	15%
D-cache tags	54417	74%	21531	29%
D-cache data	294912	100%	3481	1.2%
D-cache snoop tags	37420	87%	0	0%
Instruction trace buffer	65536	100%	0	0%
AHB trace buffer	65536	100%	N/A	N/A
Totals	864277	96%	100675	12%

5.4 Extrapolated probabilities of defects in CPU0 to evade screening

The analysis in section 5.3 shows that there is a non-zero probability for defects in CPU0 to evade screening with production test program revision 3. Since no parts tested so far (see section 5.1) have failed in the MEM5 subtest, the probability of defects in CPU0 of parts screened with revision 3 can only be extrapolated.

Since the MEM1 subtest completely screens for address decoding errors, a probability can be extrapolated based on an assumed overall stuck-at defect rate and the bit-state coverage from Table 2. An observational upper bound for the rate of at least one stuck-at defect in CPU0 memories prior to MEM1-screening is 1% (see section 5.1). This is an upper bound because the observed rate also includes distributed defects. The resulting extrapolated probabilities are given in Table 3. Numbers for CPU1 memories are included for an order of magnitude comparison. But note that the assumptions behind the calculations do not hold for CPU1 memories since the FUNC subtest does not screen out all distributed defects.

Entries in boldface indicate possibly "fatal errors" (defined as being able to cause a software crash or data corruption). It is assumed that residual defects in CPU0 memories after MEM1 screening are single-bit defects with no correlation for occurrence of multiple defects within the same data word. Under this assumption, only defects in the IU register file would cause fatal errors.

In contrast, residual errors in CPU1 memories after the FUNC subtest screening may cause fatal errors in any of the memories. This is because they may contain "distributed defects" which are fatal in all memories (apart from the Instruction and AHB trace buffers, which should not be used by applications). Again, refer to section 6.

Table 3 Extrapolated probability of defects to be present in parts screened only with MEM1 (production test program revision 3).

Memory	CPU0	CPU1
IU register file	**0.004 %**	**0.02 %**
I-cache tags	0.011 %	**0.03 %**
I-cache data	0.000 %	**0.34 %**
D-cache tags	0.027 %	**0.07 %**
D-cache data	0.000 %	**0.40 %**
D-cache snoop tags	0.008 %	**0.06 %**
Instruction trace buffer	0.000 %	0.09%
AHB trace buffer	0.000 %	N/A

Note that if a defect is present the IU register file of a particular part, then the defect is expected to be discovered immediately as soon as non-trivial software is executed on the part (e.g. during board-level unit tests).

Finally, note that 93% of the possible single-bit defects in CPU0 memories that could remain after screening with production test program revision 3 are in the caches where the LEON3FT SEU-mitigation would handle them with no functional software impact. See section 6 for details.

6 IMPACT OF A DEFECT IN A CPU MEMORY ON SOFTWARE

Terms like "single-bit" error are used in this section since they are convenient from the point of view of the error correction and detection features in the GR712RC[2]. But the term is inaccurate for errors caused by permanent defects. In this context we say that an "*n*-bit error" occurs when the data read out from a particular memory differs by *n* bits compared to what would have been read out if the same software had been run on a part without defects.

[2] Suggested reading: Section 4.8 "Error detection and correction" in the GR712RC user's manual [RD2]

By the term “stuck-at defect” we refer to a defect that causes a bit to remain at 0 or 1 despite having been written with the opposite value. For such stuck-at defects, the relation between defects and “*n*-bit errors” is direct. But memories can have other types of defects. One class is addressing errors, where writes to one address can affect what data is read out from a *different* address at a later time. In that case, a single defect may cause any number of bit errors in a data word.

By design, the GR712RC can recover from any “non-stuck” single-bit error, and this process is transparent to software. But a single-bit error caused by a stuck-at defect cannot be cleared and would be redetected every time the bit is reread. For this reason, the GR712RC cannot recover from all single-bit stuck-at errors. Furthermore, the GR712RC in general cannot recover from multi-bit errors.

An overview of possible consequences of different numbers of bit errors in different types of memories is given below. Fatal errors (defined as errors that can crash or corrupt the application) have been marked with boldface:

I. IU Register file:
   1. Transient single-bit error: IU register file contents corrected, instruction restarted after 6 cycles, error counter incremented, and CPU continues with normal operation.
   2. **Stuck-at single-bit error**: Detected by ECC, but CPU enters an infinite loop due to restarting the instruction that caused the error to be detected thereby rereading the still corrupted register.
   3. **Two-bit erro**r: Detected by ECC and triggers trap 0x20 (register_access_error).
   4. **Three or more bit errors**:
      a. **Trap 0x20** (probability[3] $40 - 70\%$)
      b. **Data incorrectly corrected**. Leads to data corruption or infinite loop (see above). (probability $30 - 60\%$)
      c. **Data incorrectly considered correct**. Execution continues uninterrupted, but with erroneous data. Unpredictable results. (probability $< 1\%$)

II. I-cache and D-cache (data and tags)
   1. Parity error (all single-bit errors and >60% of multi-bit errors): Detected by parity check, and results in a cache miss. Leads to performance degradation due to re-loads from main memory. Error counters saturate.
   2. **No parity error** (<40% of multi-bit errors): Various kinds of data corruption.
      a. I-cache data: Wrong instruction executed or trap 0x02 (illegal_instruction)
      b. D-cache data: Wrong data used. Unpredictable results.
      c. I-cache or D-cache tag
         1. Corruption of “valid”-bits in cache tag:
            a. **false valid**: wrong or invalid instruction or data
            b. false invalid: reload from external memory, continue with normal operation
         2. Corruption of address-bits in cache tag:
            a. **false hit**: wrong instruction or data
            b. false miss: reload from external memory, continue with normal operation
         3. Corruption of MMU context-bits in cache tag (MMU enabled):
            a. **false context hit**: wrong instruction or data
            b. false context miss: reload from external memory, continue with

[3] The probabilities are approximate because the exact probability depends on which model is used for defect rate per data word. However, the given probability range holds for a broad class of models.

normal operation

III. D-cache snoop tags:
   1. Single-bit error: Detected by parity bit. Cache line invalidated more often than usual, normal operation continues. Not tracked by any counter.
   2. Odd number of bit errors: Same as 1.
   3. Non-zero even number of bit errors:
      a. **false miss**: Failure to invalidate cache entry. Cache coherency lost (typically leads to process synchronization error).
      b. false hit: Additional invalidation of cache entry, continue with normal operation.

IV. FTAHBRAM: From an error management point of view, this is equivalent to an external memory. See [RD3] for details.

If a GR712RC part contains a defect capable of triggering one of the fatal errors bolded in the list above, then this would in most cases be detected almost immediately by running non-trivial software on the part. The IU register file and caches are small compared to typical application software. One second of execution of a memory intensive (relative to the 2 × 16 KiB/CPU cache size) application is enough to replace the data in the IU register file and caches multiple times. Hence most fatal defects should be detected within seconds by running software.

As shown in Table 2, the distribution of insufficiently tested bits is not uniform. The most severe IU register file and instruction cache errors are screened away by the FUNC subtest. This has the result that small software applications (relative to the cache size of 2 × 16 KiB) has a smaller chance of encountering untested bit-states than large and/or complex software applications. But when a small software application is recompiled, the register and cache access pattern may change slightly and result in untested bit-states being accessed. SMP operating systems such as RTEMS, VxWorks, and Linux have a particularly large chance of using untested bit-states. These are also sensitive to snoop tag errors.

6.1 Correctable error counters

Non-fatal errors due to defects are subtle since they are transparent to software and have characteristics akin to radiation-induced single-event upsets, except that they occur deterministically. Software that regularly reads out the cache parity error counters would quickly find that something is not normal since the applicable counter would saturate quickly even in the absence of radiation. Monitoring these counters is strongly recommended for any software intended for applications that will operate in a radiation environment.

Each CPU has a set of counters that keeps track of the number of detected errors of each type[4] that have occurred. Cache parity error counters can be read via the cache control register, and the IU register file error counter can be read via the ASR16 register (see sections 4.5.4, 4.5.6 and 4.8.2 in [RD2]).

[4] Except for snoop tag parity errors which are not tracked. A snoop tag parity error is treated the same way as if it was a snoop hit, invalidating the cache line.

A C-program can read out the cache control register and ASR16 with the following inline assembly code snippets:

```
/* Read cache control register*/
uint32_t ccr;
asm volatile (
    "lda [%1] 0x02, %0"
    : "=r"(ccr)
    : "r"(0x00)
);

/* Read ASR16 register */
uint32_t asr16;
asm volatile (
    "mov %%asr16, %0"
    : "=r"(asr16)
    :
);
```

The registers can also be conveniently read out in GRMON with the commands “`info reg -v cpuX::ccr`” and “`reg asr16 cpuX`”. An example readout of these registers for CPU1 is given below. In this case all five counters were zero.

```
grmon3> info reg -v cpu1::ccr
  LEON3FT SPARC V8 Processor
                  Cache control register                  0x008b800f
      29     rft               0x0         Register file test select
      28     ps                0x0         Parity Select
      27:24  tb                0x0         Test Bits
      23     ds                0x1         Data cache snoop enable
      22     fd                0x0         Flush data cache
      21     fi                0x0         Flush Instruction cache
      20:19  ft                0x1         FT scheme
      17     st                0x1         Separate snoop tags
      16     ib                0x1         Instruction burst fetch
      15     ip                0x1         Instruction cache flush pending
      14     dp                0x0         Data cache flush pending
      13:12  ite               0x0         Instruction Tag Errors
      11:10  ide               0x0         Instruction Data Errors
       9:8   dte               0x0         Data Tag Errors
       7:6   dde               0x0         Data Data Errors
       5     df                0x0         Data Cache Freeze on Interrupt
       4     if                0x0         Inst. Cache Freeze on Interrupt
       3:2   dcs               0x3         Data Cache state
       1:0   ics               0x3         Instruction Cache state

grmon3> reg asr16 cpu1
      asr16 = 49152 (0x0000c000)
```

There is no dedicated parity error counter for the snoop tags, so defects located in the snoop tags would be more difficult for software to detect. However, GRMON provides a set of “`dcache`”-commands that may be useful. For example, errors in the snoop tags can lead to D-cache inconsistency and GRMON provides the command “`dcache diag cpuX`” which compares all D-cache contents with external memory and finds any inconsistency.

6.2 Radiation effects

By definition, non-fatal defects allow software to operate normally with a performance reduction since they are handled by internal EDAC. However, an SEU that flips a bit in a cache word or tag that already contains a single-bit stuck-at defect is functionally equivalent to a multi-bit upset in that word. Since the GR712RC generally does not tolerate multi-bit upsets, this may lead to a crash or corruption of the application.

Suppose that there is a permanent single-bit stuck-at defect in one of the cache memories and that the overall per bit SEU rate of the cache memories is r. Generally, each bit is part of a group of 9 bits over which parity is computed and an undetectable error only happens if one of the other 8 bits is flipped by an SEU. Hence the rate of undetectable errors is $8 \times r$ for the cache word containing the stuck bit. However, there are exceptions:

- In the snoop tags, one parity bit is used for 20 data bits => $20 \times r$
- Instruction tags are 40 bits long, but still only use four parity bits. The fourth parity bit covers both the 8-bit MMU context and 8 address bits => $16 \times r$
- The first two parity bits in the D-cache tag field only cover 4 data bits each. => $4 \times r$

A representative rate for the GR712RC caches in a GEO environment is $r = 2 \cdot 10^{-7}$ SEU/bit/day[5]. The undetected upset rate with a single stuck-at bit somewhere in the caches is therefore at most $20 \times r = 4 \cdot 10^{-6}$ events/day. A mean time to upset of above 680 years. See also [RD1] for comparison.

Total ionizing dose mainly causes drift of electrical parameters. And, as stated in section 5.1, no temperature or voltage dependence of functional errors has been observed. Therefore, new functional errors are not expected at end of life due to total radiation dose.

7 SOFTWARE PACKAGE FOR BOARD-LEVEL MEMORY TESTS

Upon request to support@gaisler.com, GR712RC customers that have parts screened with revision 3 of the production test program already integrated into boards will be provided with a software package “gr712rc-tn0002-sw”. The software package includes a prebuilt binary, C and assembler source code, and a TCL script that can be used with GRMON to run the tests. Detailed instructions can be found in the README included in the software package. The rest of this section provides an overview of the contained test software.

The software is intended to be loaded, started and monitored by GRMON2 or GRMON3 connected via a JTAG or SpaceWire debug link. The software executes out of the FTAHBRAM (On-chip Memory with EDAC Protection) so there is no dependence on external memory.

The software and GRMON will only use the FTAHBRAM, CPU0, CPU1, DSU (Hardware Debug Support Unit), IRQMP (Multiprocessor Interrupt Controller) and GPTIMER/GRTIMER (General Purpose Timer Unit) cores. No accesses to the FTMCTRL (Fault Tolerant Memory Controller) memory regions (PROM, IO, SRAM and SDRAM) or IO peripherals are made.

The software tests the full IU register file, I-cache (tags and data), and D-cache (tags, data, and snoop

[5] The GR712RC caches and IU register file consist of 384290 bits/CPU. So, the given per-bit rate corresponds to an overall upset rate of $384290 \times 2 \cdot 10^{-7} \approx 0.08$ errors/CPU/day. This is comparable to the example value $5.8 \cdot 10^{-7}$ errors/CPU/s (or 0.05 errors/CPU/day) given in [RD1]. Relatedly, [RD1] states that the GR712RC caches and IU register file consist of 432128 bits/CPU instead of 384290. The difference is due to the former counting the number of bits in all memory macrocells, whereas the latter only counts the number of bits that are actually used. For example, the register file is implemented using two 256x40 macrocells, but only 135 words are ever accessed and only 39 out of 40 bits per word are used. For an effective size of 2x135x39.

tags) of both CPUs. Parity and ECC bits are also covered by the test. Optionally, the software can also be configured to test the CPU1 instruction trace buffer but note that this memory should not be used by application software. The software does not test for occurrence of the FTAHBRAM back-to-back write errors described in section 4.

A review of historical production test data of GR712RC parts shows that if defects in CPU memories are detected at one supply voltage and temperature condition, then defects will be detected in the entire operating range of the part (see section 5.1). Therefore, it is sufficient to run the software at room temperature and nominal supply voltage to determine if defects are present or not.

It is possible to tailor the software such that it can be loaded and executed on boards where no GRMON debug link is available. However, the software is not suitable for permanent integration into application software.

7.1 Memory test methodology

The baseline algorithm used in the memory tests has two parts. First an address correctness step where unique values are written into each word of the memory block. Next a MARCH-C pattern to find defective bits. Here, it is verified that each bit can be written with both 0 and 1. The exact details vary in each type of memory, depending on if the memory a single- or dual-port type, and on the various kinds of automatic ECC/parity computation, as well as available diagnostic memory access interfaces.

A pseudo-code description of the test steps follows below:

```
## Baseline address correctness algorithm:
NOTE: The total size of the memory in number of words is denoted N
-------------------------------------
1. Initial write
for i=0 to N-1
    write i to address i

2. Readback, second fill and final readback
for i=0 to N-1
    j = N-1 - i
    read from address i and compare value against i
    write j to address i
    read from address i and compare value against j

## Baseline MARCH-C pattern
NOTE: The word width is between 21 and 40 bits depending on memory type.
NOTE: In some test routines, the readback is divided into three separate loops where
ECC/parity bits, odd word addresses, and even word addresses are read separately.
-------------------------------------
1. Initial write
for i=0 to N/2-1
    write 0x55...55 to address 2*i+0
    write 0xAA...AA to address 2*i+1

2. Readback and write bitwise inverse
for i=N/2-1 downto 0
    read address 2*i+1 and compare against 0xAA...AA
    write 0x55...55 to address 2*i+1
    read address 2*i+0 and compare against 0x55...55
    write 0xAA...AA to address 2*i+0

3. Readback of inverted pattern
for i=0 to N/2-1
    read address 2*i+0 and compare against 0xAA...AA
    read address 2*i+1 and compare against 0x55...55
-------------------------------------
```

7.2 Example output

Below is example output of the software when executed on a GR712RC part with no errors in either CPU0 or CPU1 memories.

```
$ grmon -ftdi -nosram -nosdram -c systest.tcl

  GRMON debug monitor v3.2.14 64-bit pro version

  a0000000                          192.0kB / 192.0kB   [===============>] 100%
  Finished washing!
          A0010000 .text             28.8kB /  28.8kB   [===============>] 100%
          A0017320 .rodata             16B              [===============>] 100%
          A0017330 .data              200B              [===============>] 100%
  Total size: 28.99kB (776.16kbit/s)
  Entry point 0xa0010000
  Image .../systest-cpu0 loaded
  CPU 0:  Program exited normally
  CPU 1:  Power down mode
  0xa0017430  00000001  00000001  00000000  00000000    ................
  0xa0017440  4f4b2120  4f4b2120  4f4b2120  ffffffff    OK! OK! OK! ....
  0xa0017450  ffffffff  ffffffff  ffffffff  ffffffff    ................
  0xa0017460  00000000  00000007  ffffffff  ffffffff    ................
  a0000000                          192.0kB / 192.0kB   [===============>] 100%
  Finished washing!
          A0010000 .text             28.8kB /  28.8kB   [===============>] 100%
          A0017320 .rodata             16B              [===============>] 100%
          A0017330 .data              200B              [===============>] 100%
  Total size: 28.99kB (778.70kbit/s)
  Entry point 0xa0010000
  Image .../systest-cpu1 loaded
  CPU 0:  Program exited normally
  0xa0017430  00000001  00000001  00000001  00000000    ................
  0xa0017440  4f4b2120  4f4b2120  4f4b2120  ffffffff    OK! OK! OK! ....
  0xa0017450  ffffffff  ffffffff  ffffffff  ffffffff    ................
  0xa0017460  00000000  00000007  ffffffff  ffffffff    ................

  ################################################################
  Result:
  Test CPU0 IU-Registers     OK!
  Test CPU0 I-Cache          OK!
  Test CPU0 D-Cache          OK!
  Test CPU1 IU-Registers     OK!
  Test CPU1 I-Cache          OK!
  Test CPU1 D-Cache          OK!

  CPU0 OK!
  CPU1 OK!

grmon3>
```

Information furnished by Cobham Gaisler is believed to be accurate and reliable. However, no responsibility is assumed by Cobham Gaisler for its use, or for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Cobham Gaisler.

All information is provided as is. There is no warranty that it is correct or suitable for any purpose, neither implicit nor explicit.