System z Redundant Array of Independent Memory

Dave Hayslett
hayslett@us.ibm.com

IBM SWG Competitive Project Office

8 April 2011

The New zEnterprise Memory System – Another System z Innovation!

System z is the most reliable server on the planet, and it has achieved this distinction by pursuing a multi-layer strategy for reliability and serviceability. First, we use quality components and manufacturing processes. Next, we test extensively, with deep burn-in. In the next layer we build in redundancy to provide seamless error detection and correction. Good examples are the spare CPUs that can replace any failed CPUs without loss of data. Memory subsystems have extensive error correction redundancy. At the next layer, we build in serviceability, with an independent service element that initiates self repair or phones home for a service call. Our software is mature and well tested to reduce software errors. In the next layer, Sysplex clustering technology enables entire system failovers without loss of data. Finally, our automated disaster recovery ensures continued business operation in the event of catastrophic failures.

These innovations are the result of years of focused improvements designed specifically to cope with failure modes and scenarios. This article examines the latest innovation designed to deliver the most reliable memory subsystem ever created. We will demonstrate how System z provides 26 bytes of redundant error detecting and correcting memory bits for every 64 bytes of data, and show how it furthers IBM's commitment to System z's five nines availability.

What is RAIM and why is it important?

The IBM zEnterprise system introduces a revolutionary feature, the Redundant Array of Independent Memory (RAIM) subsystem design as a standard feature on all zEnterprise Servers. RAIM is an error correction scheme that is conceptually similar to the well-known RAID technology in disk drives. Complex components, like DIMMs and buffer chips, can sometimes fail. Increases in memory density can drive these failure rates even higher. (Even stray cosmic and alpha rays can cause transient errors that must be detected and corrected.) Failure of any of these components can have a large impact on system availability. RAIM was introduced to address these scenarios.

In addition to correcting against DRAM and control bus failures, RAIM also protects the server from single-channel errors such as sudden control, bus, buffer, dual in-line memory module (DIMM), and massive DRAM failures, thus achieving the highest System z memory availability to date.

Various techniques have been attempted over the years to address memory reliability. Certainly, improvements such as symbol ECC, bus CRC, scrubbing, and sparing have helped with typical DRAM and bus failures. However, every once in a while there can be more severe errors that can't be dynamically repaired by those methods.

One technique that has been introduced recently to address these types of errors is memory mirroring. This technique is useful as it provides protection of the same types of memory events as RAIM. However, protecting the entire memory space requires installed memory to be doubled, and can be prohibitively expensive. If more than 25% of memory needs to have this added protection, RAIM is a more cost-efficient approach than memory mirroring. It is also very difficult to reliably predict which parts of memory should be protected by mirroring, and which parts shouldn't be protected. RAIM allows for an efficient way to cover all of memory.

Other RAID-like techniques have been introduced, including some hot-plug options. However, they are very limited on the memory capacity allowed with these systems due to the need to allow physical access to the DIMMs that support hot-swapping. On zEnterprise, concurrent repair is done on a book level which allows a much denser memory footprint with less plugging risk. This allows for a lower-risk, higher-capacity, higher-performance memory subsystem.

Together these improvements are designed to deliver ***System z's most resilient memory***

subsystem to date.[1] Additionally, RAIM's fully-integrated operation imposes **no performance penalty** on the operation of System z servers.

Why is error correction important? When memory failures exceed the capabilities of the memory correction system, the result is a system crash. As memory becomes more dense and as overall memory capacity increases, there are higher probabilities of errors that come up and more robust corrections schemes are necessary. (On Windows machines, uncorrected memory failures are one of the many causes of the infamous “Blue Screen of Death”.)

How does RAIM work?

In order to implement RAIM, 'extra' memory is installed on each book. Each z196 book can contain up to 960 GB of physical memory, for a total of 3840 GB (3.75 TB) of installed memory per system (with four books). 20% of the physical installed memory is used to implement the RAIM design, resulting in up to 768 GB of available memory per book and up to 3072 GB (3 TB) per system. [2] RAIM, and this additional memory, is provided as a standard feature of the z196.

Each book has from 10 to 30 DIMMs (depending on how much memory is installed). The DIMMs are connected through three memory control units (MCUs), each located on a processor unit on the book. Each MCU uses five channels, one of them for RAIM implementation, on a 4 +1 (parity) design. Each channel has one or two chained DIMMs, so a single MCU can have five or ten DIMMs. The parity of the four “data” DIMMs are stored in the DIMMs attached to the fifth memory channel. This data, along with CRC bus protection in each of the five channels allows for failures in a memory component to be detected and corrected dynamically. Figure 1 illustrates the memory layout of a fully-configured book.

The RAIM design detects and recovers from DRAM, socket, memory channel, or DIMM failures. It is loosely similar to a RAID level 3 design. Five memory channels are involved in any read or write request. (A memory channel is either one DIMM, or two DIMMs chained together.) Cyclic Redundancy Check (CRC) data is used on each channel bus to detect and isolate channel errors. On top of that, RAIM ECC is used to cover the entire end-to-end path from original store to the DIMMs to the final fetch. During reads, this CRC data is verified again, allowing for proactive error detection and correction. This allows the recovery from multiple chip failures that RAIM provides. Figure 2 illustrates how data is written to memory and read back using RAIM.

1 zEnterprise IBM US announcement letter http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&supplier=897&letternum=ENUS110-170.

2 IBM zEnterprise System Technical Guide, SG24-7833-00.

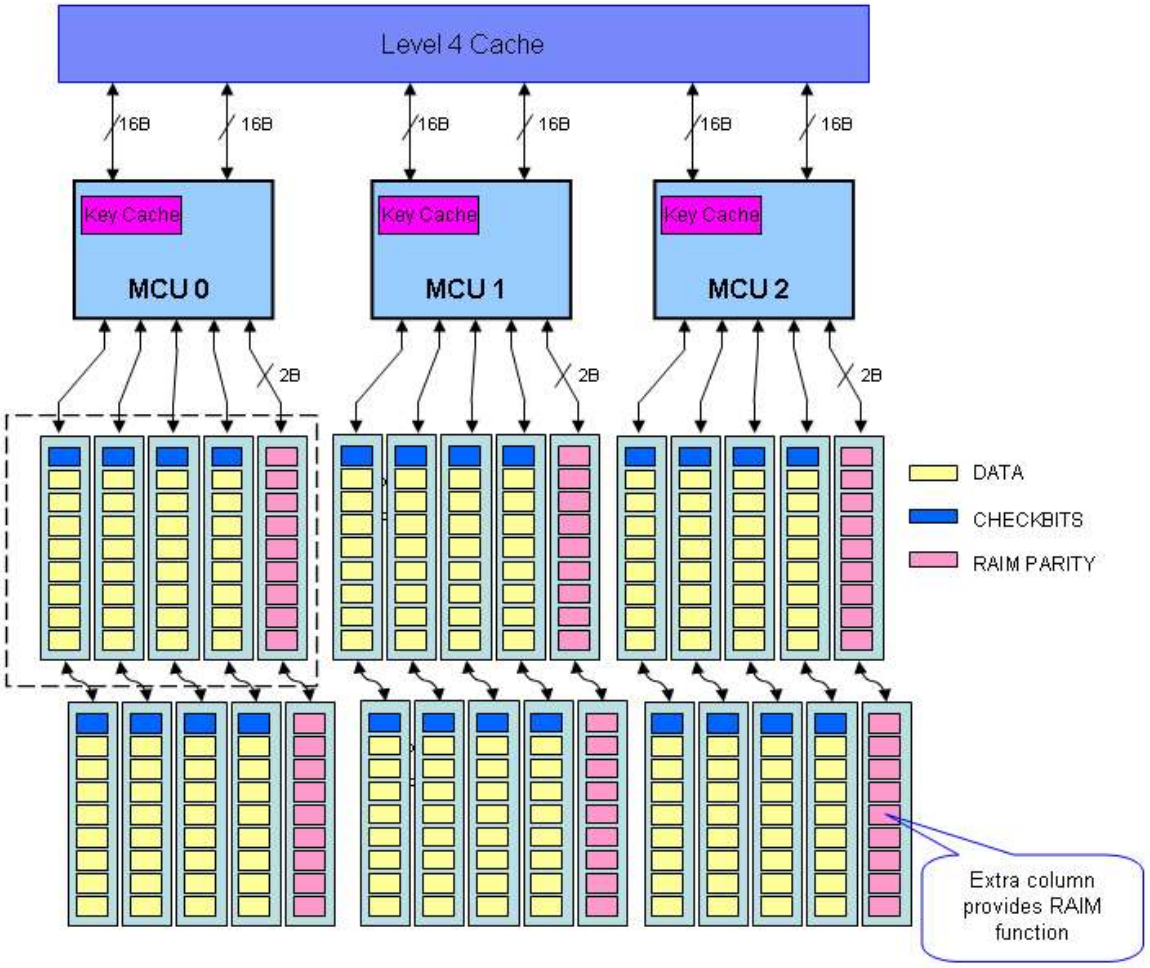

Figure 1: Memory layout of a fully-configured z196 book.

DRAM and channel marking techniques are employed in the RAIM design, eliminating the need for DRAM sparing. DRAM marking eliminates the complexity of copying over chip data from one chip to another. Once a chip is known bad, it is marked as bad to the ECC code and the code automatically ignores all content from that DRAM.

Likewise, a channel mark can be applied to one failing channel that is deemed to be unreliable. Once a channel mark is applied to a channel, all data from that channel is effectively ignored, even if there is another failure in another channel. For DRAM-only failures, data is corrected using in-line ECC correction. In the event of the detection of a memory control error with accompanying CRC violations, a 3 tiered recovery sequence is started; this can result in actions ranging from the dynamic correction of soft errors, bus data lane sparing, bus clock lane sparing, up to marking an entire channel bad and causing a service request to be issued.

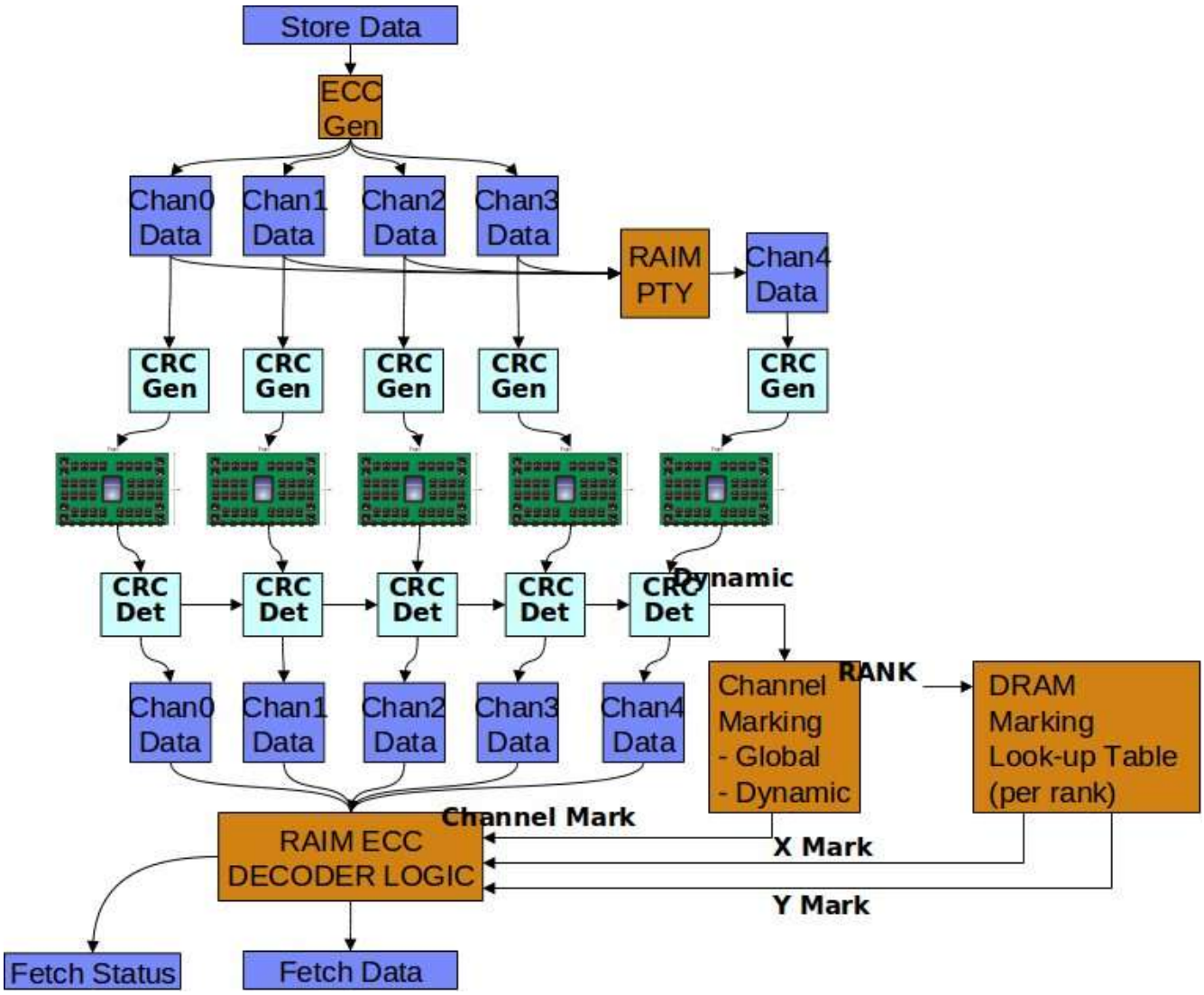

Figure 2: RAIM store and fetch operations

Table 1 summarizes the RAS features implemented by the RAIM subsystem.

RAS Feature	Description
RAIM ECC	Five-channel ECC which can detect and correct new DRAM failures as well as most varieties of single channel failures in the memory subsystem.
DRAM chip marking	Up to two DRAM chip marks can be applied per rank in order to ignore errors from known defective DRAM chips. Unlike DRAM chip sparing, these marks can be applied without having to replicate any chip data.
Channel marking	Channel marking is the ability to designate one of five RAIM channels as defective. The channel mark provides 100% correction of the data in the ignored channel. There are four levels of channel marking: dynamic, tier3, temporary, and permanent.
CRC Bus Detection	Upstream and Downstream channels are checked using CRC.

RAS Feature	Description
Tier1 reset	Tier 1 recovery quiesces the channels, resets memory channel resources, and then resends stores that may have been dropped.
Tier2 data calibration Lane sparing	Tier 2 recovery recalibrates memory data buses and spares out bad data lanes.
Tier3 clock calibration Lane sparing	Tier3 recovery recalibrates memory clocks and spares out bad clock lanes. Firmware performs fastscrub to clean-up stale data.
Scrubbing	Scrubbing is the process of periodically reading, correcting, and writing back memory to correct soft errors. Scrubbing provides chip error counts which are used to apply DRAM chip and channel marks.
Service Request	A service request is an event which requests a part replacement. Some examples of memory-related service requests include: - Permanent, full channel RAIM degrade. - Overflow of the DRAM mark capabilities within a rank. - Overflow of bus spare lanes within a channel or cascade.

Table 1: RAS features implemented by the RAIM subsystem

RAIM is a powerful new innovation in mainframe memory that furthers the IBM System z's world-class reliability and serviceability by providing extraordinary protection against potential failures in the memory subsystem.

Reference

For more detail on RAIM and other zEnterprise topics, please see “IBM zEnterprise System Technical Guide”, SG24-7833-00, available from http://www.redbooks.ibm.com. Also watch for the upcoming in-depth article “zEnterprise RAIM Memory Subsystem” from which much of this material is drawn.

The detailed mathematics behind RAIM are described in “A new class of array codes for memory storage”.

*P. J. Meaney, L. A. Lastras-Montaño, V. Papazova, E. Stephens, J. Johnson, L. Alves, J. O’Connor, W. Clarke, “zEnterprise RAIM Memory Subsystem”, IBM J. Res & Dev., January, 2012.

*L. A. Lastras-Montaño, P. J. Meaney, E. Stephens, B. M. Trager, J. O’Connor, L. C. Alves, "A new class of array codes for memory storage", in Proceedings of the Information Theory and Applications Workshop, University of California at San Diego, February 6-11, 2011 (Invited Paper).