IBM zEnterprise redundant array of independent memory subsystem

P. J. Meaney
L. A. Lastras-Montaño
V. K. Papazova
E. Stephens
J. S. Johnson
L. C. Alves
J. A. O'Connor
W. J. Clarke

The IBM zEnterprise® system introduced a new and innovative redundant array of independent memory (RAIM) subsystem design as a standard feature on all zEnterprise servers. It protects the server from single-channel errors such as sudden control, bus, buffer, and massive dynamic RAM (DRAM) failures, thus achieving the highest System z® memory availability. This system also introduced innovations such as DRAM and channel marking, as well as a novel dynamic cyclic redundancy code channel marking. This paper describes this RAIM subsystem and other reliability, availability, and serviceability features, including automatic channel error recovery; data and clock interface lane calibration, recovery, and repair; intermittent lane sparing; and specialty engines for maintenance, periodic calibration, power, and power-on controls.

Introduction

The IBM zEnterprise* 196 (z196) represents a significant leap forward in reliability and availability of the memory subsystem. This new memory design, combined with the microprocessor and cache subsystem enhancements, achieves higher capacity, higher performance, and higher reliability, compared with previous designs [1].

Memory size and density have continued to increase substantially as computer systems have become more powerful. The probability of encountering a memory failure has increased proportionally. New and innovative techniques to protect against ever greater susceptibility to hard and soft failures have evolved into an elaborate science over the past several decades. The IBM zEnterprise redundant array of independent memory (RAIM) design is the result of this ongoing research and development.

Early computers used parity to provide detection of memory errors; however, they lacked any correction capability. Richard Hamming, an early pioneer in the field of error-correction codes (ECCs), took simple parity and extended it not only to detect but also to correct errors. This was accomplished with the addition of an ECC field to each data word [2].

The more powerful Reed–Solomon ECCs were first introduced by Irving Reed and Gustave Solomon [3]. These and other symbol-oriented codes have been used by IBM and others to improve memory reliability. This more advanced ECC technology is capable of detecting, isolating, and correcting full memory chip failures and memory bus failures [2].

In addition to ECC, other techniques have been used to improve memory reliability. For instance, scrubbing was added to periodically read and write memory to correct soft errors. In addition, spare dynamic RAM (DRAM) chips were introduced so that a faulty chip could be logically replaced by a spare chip, thus avoiding a service call [4].

Other innovations such as memory channel protection, memory migration, dual in-line memory module (DIMM) sparing, and memory channel hot plug have been also deployed to improve memory availability. Although these improvements are worthwhile, none of them can detect and correct for a catastrophic DIMM failure. Mainframe servers must operate unimpaired, even in the presence of complete DIMM failures. Methods have been attempted using redundant array of independent disk (RAID) techniques [5].

RAID-1 memory mirroring, in conjunction with standard ECC, has been offered as a feature to allow a system to survive more catastrophic full DIMM failures [6]. Although this technique is plausible, it can be prohibitively expensive when applied to the entire memory space.

Digital Object Identifier: 10.1147/JRD.2011.2177106

0018-8646/12/$5.00 © 2012 IBM

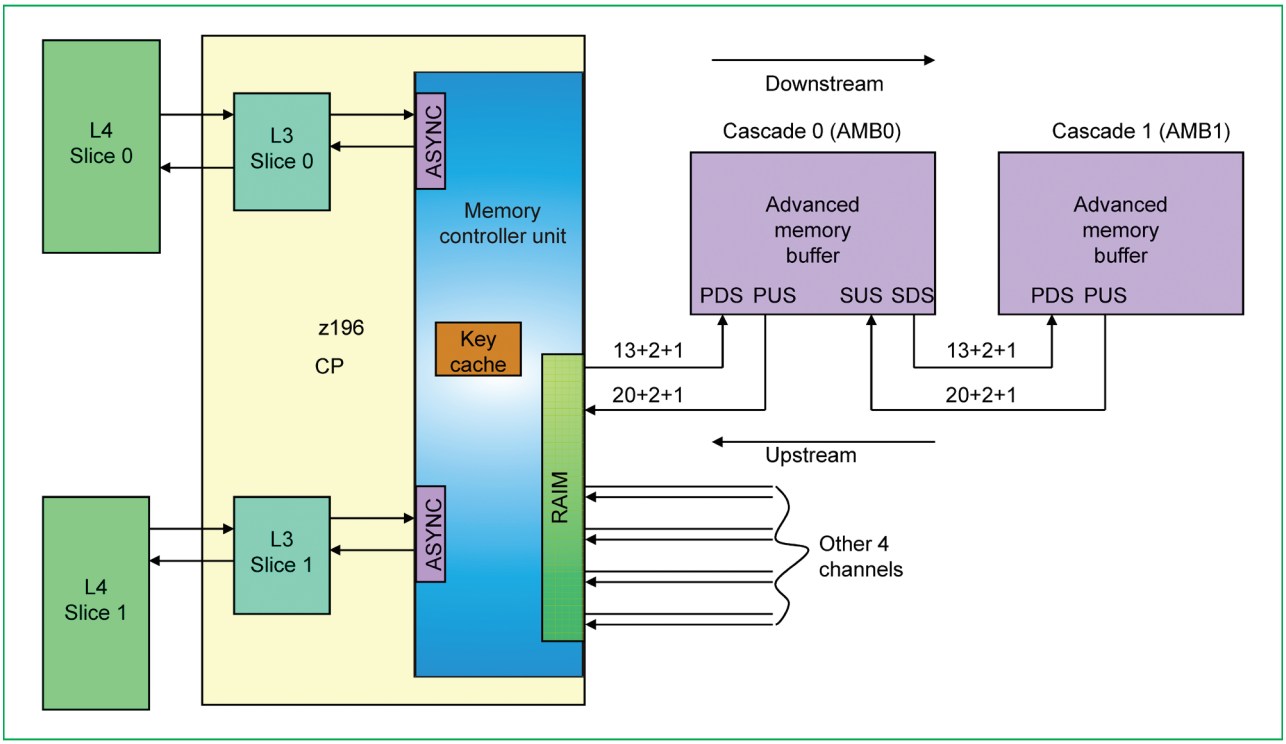

Figure 1

zEnterprise memory subsystem. Two-cascade example (one channel shown). (CP: central processor; RAIM: redundant array of independent memory; PDS: primary downstream receiver; SDS: secondary downstream driver; PUS: primary upstream driver; SUS: secondary upstream receiver; AMB0: advanced memory buffer on cascade 0; AMB1: advanced memory buffer on cascade 1.)

Another possible method is to adapt a RAID-4/5 scheme to a memory setting by combining individual channel-based ECC with an additional redundant channel. In this scheme, individual cache lines are stored in a single channel per DIMM. This scheme has significant performance challenges, requiring two read–modify–write actions for every write. Additionally, this method can be less effective at detecting catastrophic memory failures. A RAID-3 scheme may be used instead, where cache lines are striped across a number of memory channels [7]. The design that we present in this paper is significantly more efficient in terms of the redundancy needed to correct and detect the various kinds of failures that we address.

Accurately detecting and preventing data corruption while also maintaining system availability is undeniably paramount when considering computer systems running business-critical applications.

At the heart of the zEnterprise memory design is RAIM, which is described below. This subsystem is a comprehensive hardware and firmware solution that provides detection and correction for channel-level failures as well as full DIMM and DRAM failures. It is implemented using a sophisticated memory controller, an advanced memory buffer (AMB) chip, and system host firmware. This system features innovative RAIM ECC codes, bus cyclic redundancy code (CRC) detection, a three-tiered recovery system, two DRAM chip marks per rank, four levels of channel marking, dynamic lane sparing, and intermittent lane support to provide the most robust memory subsystem seen on System z. RAIM is standard on all zEnterprise models.

Memory subsystem

The zEnterprise memory subsystem is shown in **Figure 1**. Fetch, store, and cast-out requests originate from the Level 4 (L4) cache level, which is sliced into even and odd double-word addresses. These requests are forwarded to the appropriate central processor chip through the appropriate L3 controller and eventually through an asynchronous interface to the memory controller unit (MCU) in charge of the target data or key memory space [8].

The MCU controls a set of memory DIMMs that are organized into five channels. There are three MCUs per node. Each MCU has five memory ports, each of which can communicate with a memory channel using unidirectional, upstream, and downstream differential buses.

The channel bus, including data and clock signals, employs exclusively differential input/output (I/O) signaling techniques. Although differential I/O helps to provide better signal integrity for higher bus frequencies, it is costly when it comes to wires and I/O used for fetches and stores. Therefore, CRC is used to detect bus failures.

As shown in Figure 1, the downstream bus consists of 32 wires making up 16 logical lanes, i.e., 13 data lanes, 1 clock lane, and 2 spare lanes (used to repair two data lanes or one clock lane and one data lane). Downstream buses are sent from the secondary downstream driver of one chip to the primary downstream receiver of the next chip. The upstream bus consists of 46 wires making up 23 logical lanes, i.e., 20 data lanes, 1 clock lane, and 2 spare lanes (used to repair two data lanes or one clock lane and one data lane). Upstream buses are sent from the primary upstream driver of one chip to the secondary upstream receiver of the previous chip.

The zEnterprise uses an IBM DIMM, which is a fully buffered DIMM using a next-generation IBM-proprietary AMB chip bridging between the MCU and the DRAM chips. The AMB chip is discussed in a related article in this issue [9]. Double-data-rate-3 (DDR-3) 800-MHz x8 DRAM chips are used in the IBM DIMM. The DIMMs are divided into ranks, which are used to manage fetch and store operations, refresh, scrubbing, and power. Up to two IBM DIMMs can be cascaded in series to form a memory channel. AMB0 is the first cascade buffer chip, and the second cascade buffer chip is AMB1. Each MCU reads and writes 256-byte cache lines from a rank of each of five DIMMs across five memory channels simultaneously using a RAIM configuration.

The zEnterprise supports 4-, 8-, 16-, and 32-GB DIMMs, which are plugged in groups of five DIMMs. Within a node, all first DIMM cascades for each MCU are plugged prior to plugging any second cascade DIMMs to best balance the system and optimize performance. The maximum memory capacity of a zEnterprise high-end (HE) four-node system is 3 TB.

The zEnterprise MCU has two independent clock domains, namely, nest clock and memory clock domains, with an asynchronous interface between them. The nest domain runs at 1.3 GHz (which is a 4:1 ratio to the core clock). The memory clock domain (1.2 GHz) synchronously runs with the AMB clocks and DRAMs. The memory channel bus has double data rate. The AMB chips have an internal clock ratio of 6:1. This means that the DRAMs run at 800 MHz, whereas the cascaded buses operate at 4.8 gigabits per second (Gb/s).

Summary of RAS features

Table 1 shows a brief summary of the reliability, availability, and serviceability (RAS) features described in this paper. These are described in more detail in the following subsections.

RAIM ECC

At the heart of the MCU is the RAIM dataflow. The RAIM ECC has been custom designed to satisfy particular requirements set by the zEnterprise memory subsystem architecture. These requirements include the following:

- Up to two DRAM chip marks per rank.
- Support both x4 and x8 DRAMs.
- Correction of an entire memory channel, even in the presence of prior bad DRAM chips.
- A constant low decoding time for all failure modes, including new channel failures.
- Use of standard ECC DIMM 72-bit data I/O configurations.
- No loss of memory capacity on a channel degrade.

RAIM ECC implementation

A new class of ECCs that we call diff-MDS (i.e., maximum difference separable) has been specifically developed for System z* [10]. The memory system is organized so that five memory channels are used in any given read or write request. As shown in **Figure 2**, 64 bytes of stored data is used to generate check bits in the first four channels. The fifth channel is a strict "EXCLUSIVE OR" (XOR) of the corresponding data from the first four channels. After the data from the five channels is formatted into frames, CRC is added to each channel independently. That CRC is used to detect whether there are any downstream or upstream bus errors. If the AMB detects bad CRCs on any data bus, recovery will be invoked, first in that channel and eventually in the memory subsystem. (see Tier 1, Tier 2, and Tier 3 recovery.) If the CRC is clean, the data will be stored. Later, the data will be fetched, CRC will be checked per channel, and RAIM ECC will be used to detect and correct errors.

In the parlance of the RAID [5] literature, the implemented RAIM scheme may be loosely associated with RAID L3 where a channel takes the role of the disk, although the technologies and system considerations used to implement each are different.

Mathematically speaking, RAIM ECC is a {45, 32} symbol code where each symbol is composed of 16 bits. This means that 32 symbols are data (64 bytes in total) augmented with 13 redundant symbols (corresponding to 26 bytes). The first four channels each store a total of nine symbols (eight data and one redundant), and the fifth redundant channel stores a simple parity of the other four channels.

The symbol mapping is shown in **Figure 3**. Raw data are represented in yellow, ECC check bits are shown in blue, and the redundant RAIM channel bits are shown in orange. The ECC is capable of locating and correcting, with overwhelmingly high probability, channel failures independently of the individual channel CRC fail

Table 1 Summary of RAS features.

RAS feature	*Description*
RAIM ECC	Five-channel ECC that 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, Tier 3, temporary, and permanent.
CRC bus detection	Upstream and downstream channels are checked using CRC.
Tier 1 reset	Tier 1 recovery quiesces the channels, resets memory channel resources, and then resends stores that may have been dropped.
Tier 2 data calibration Lane sparing	Tier 2 recovery recalibrates memory data buses and spares out bad data lanes.
Tier 3 clock calibration Lane sparing	Tier 3 recovery recalibrates memory clocks and spares out bad clock lanes. Firmware performs fast scrub 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 that are used to apply DRAM chip and channel marks.
Service request	A service request is an event that requests a part replacement. Some examples of memory-related service requests include the following: • Permanent, full-channel RAIM degrade. • Overflow of the DRAM mark capabilities within a rank. • Overflow of bus spare lanes within a channel or cascade.

information, which may be further incorporated in decoding as described later. Notably, this is accomplished without the use of stored per-channel check bits. Instead, the check bits stored in the blue chips depend on all four channels, resulting in an optimal design, providing maximal error correcting and detecting power across the five channels, given the available redundancy for a large class of error scenarios of interest. Details of the mathematical algorithm can be found in [10]. Usage of this type of design was important in order to meet the design goals without requiring nonstandard DIMM configurations.

DRAM marking

RAIM has been carefully crafted so it can mark up to two DRAM chip failures per rank. While in previous generations of System z servers, DRAM sparing provided relief for failing DRAMs, marking allows for a simpler quicker switchover for failures. There is no need to wait for replication from the failing chip to the spare chip. In addition, with a RAIM design that uses sparing, some scenarios that involve a failing DRAM in one channel that gets spared by a DRAM in another channel would cause erroneous multiple-channel failures. DRAM marking eliminates these possibilities.

Channel marking

Similar to DRAM marks, the RAIM ECC design also provides the ability for channel marks. A channel mark will allow an entire defective channel (i.e., one or two cascaded DIMMs) to be defective without influencing the ECC correction of the remaining channels. Channel marking can be applied permanently (when a DIMM is defective and needs replacement) or temporarily.

The temporary channel mark can be applied during Tier 3 recovery (when a channel is replacing a clock lane and needs time for clock recalibration) or dynamically during Tier 1/2 recovery (when a soft, intermittent, or solid bus or control error occurs). In order to protect the data better as the system continues to access memory, the channel mark is dynamically applied to temporarily mark the failing channel until recovery has been completed. This combination of CRC with channel marking provides a very robust way to protect against most DIMM errors, even those that can occur without warning.

Marking summary

A summary of the marking and correction capabilities of the zEnterprise RAIM ECC is shown in **Table 2**. The columns represent previous marking states (either no DRAM

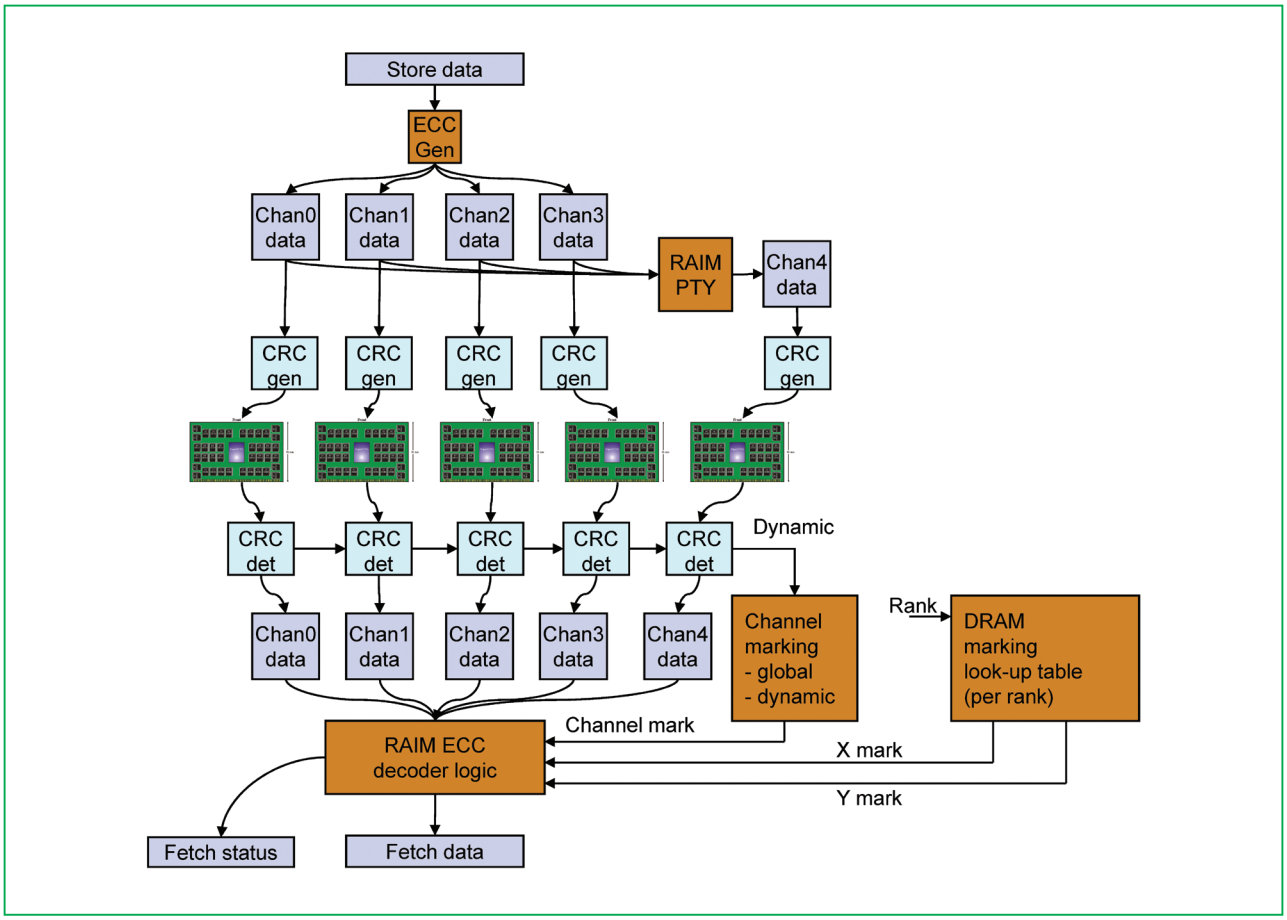

Figure 2

RAIM ECC design static channel and DRAM marking and dynamic CRC marking. (gen: generation; det: detection; PTY: parity.)

marks or one or two DRAM marks). The rows represent new errors that have not yet invoked marking. If there is a permanent channel mark, the machine will continue to run but in a service-requested state that will notify the support team that the DIMM needs to be replaced (see the last column in Table 2).

The first two rows in the table represent typical scenarios that were similarly handled on both z10* and zEnterprise. If a single DRAM in a rank fails, a DRAM mark (spare chip on z10) is applied to that chip and the CRC errors from that DRAM go away. A second DRAM in the rank can also fail, and another DRAM mark (i.e., spare) is applied. When a third DRAM fails in the rank, a service request is made to replace the failing DIMM.

However, zEnterprise can also handle other cases, as shown in the last two rows in Table 2. Two DRAMs on the same DIMM failing at simultaneously or back-to-back (before a DRAM mark is applied) can be fully corrected. In addition, if there is a full DIMM control problem, causing CRC errors to come up on the interface, zEnterprise can fully correct these errors using dynamic channel marking. Another enhancement for zEnterprise is the ability to use a channel mark during the service request call to mark a third DRAM.

Memory channel RAS

This section describes the RAS features associated with each memory channel. These features combine the AMB three-tiered recovery per channel with the five-channel RAIM ECC to achieve an integrated robust scheme of error detection, correction, and repair.

Calibration (TS0, TS2–TS7), slow reinitiation, and fast reinitiation

During power-on, the cascaded DIMM interfaces are calibrated. Interface calibration is necessary to allow the hardware to account for variations in delay and voltage due

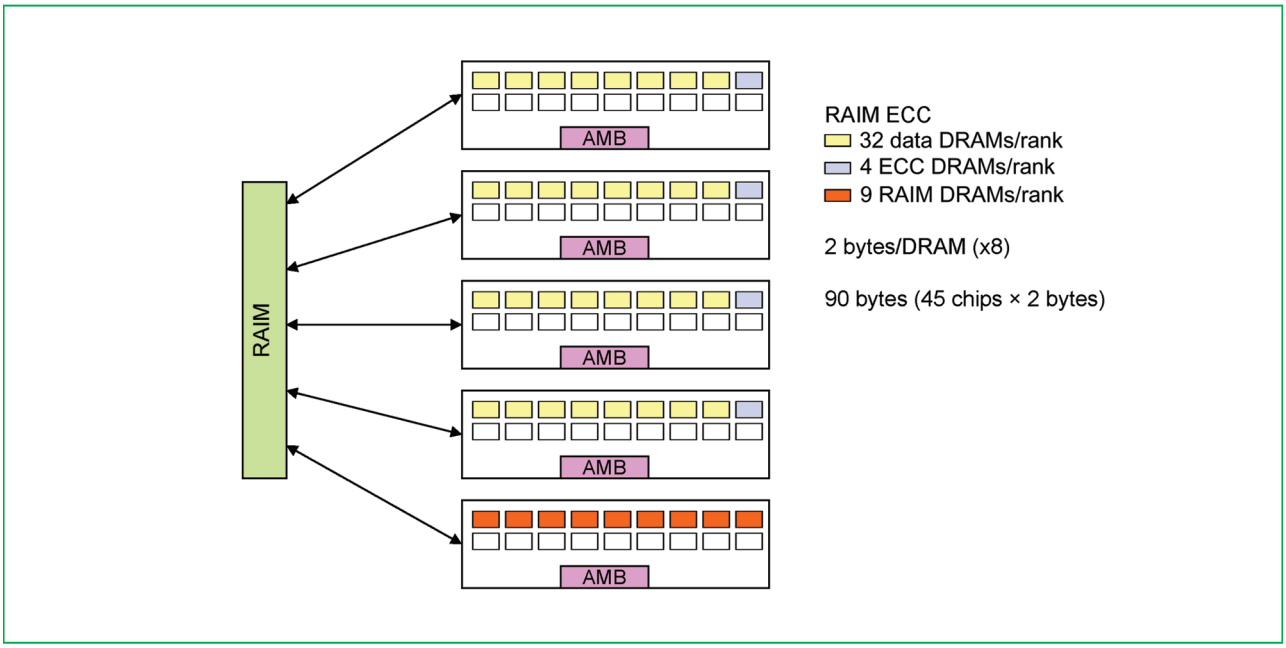

Figure 3

RAIM ECC DIMM mapping for a single 64-byte logical ECC word.

Table 2 RAIM ECC DRAM and channel marking summary. CE: correctable error. (Reproduced, with permission, from [1]; © IEEE 2010.)

Marks/new errors	*No marks*	*Single chip marked*	*Two chips marked*	*Third chip/channel mark*
None	Good	Good	Good	
One chip	CE	CE	CE	Service request
Two chips, same channel	CE	CE	CE	
Full channel (CRC errors)	CE	CE	CE	Call home for part replacement

to impedance and capacitance variation across the bits. Particular steps are required for the interface hardware to self-calibrate. These steps are referred to as training states (TS0–TS7). First, the bus clocks need to be established, both downstream through both cascades and then upstream back to the MCU. Clock training is accomplished through slow initialization (TS0). Later, the AMB is customized using TS1. Then, the data is calibrated downstream (TS2), then upstream (TS3), followed by frame locking and alignment steps (TS4–TS7). After this calibration, the interface is fully synchronized across all five memory channels. These training states, i.e., TS0 for clocks and TS2–TS7 for data, are reused during Tier 2 and Tier 3 recovery, as described later.

CRC protection

Both upstream and downstream data buses are covered by CRC detection. In the event of a single bus lane failure, the CRC is used to detect the error. Detection effectiveness is improved via interface scrambling so that, even when the bus is idle, the actual physical data is always changing based on a predetermined linear feedback shift register (LFSR) code, which is encoded and decoded on the data. This way, even if an AMB stops running, the receiving side

of the channel interface will detect a mismatch on the scrambling pattern and detect CRC errors.

CRC is also used as a communication vehicle. If a bad result or a catastrophic error is detected within one of the AMBs, it will force an upstream "poison CRC" (a known pattern of a bad CRC) back to the MCU. Each error checker has been optimally programmed to cause CRC/recovery or to be information only, depending on the impact the error has on the subsystem RAS.

As depicted in Figure 2, downstream data are sent in frames that are 13 lanes wide and 12 beats deep, for a total of 156 bits of data plus ECC plus CRC. Upstream data are sent in frames that are 20 bits wide and 8 beats deep, for a total of 160 bits of data plus ECC plus CRC.

Bit error rates

Soft and transient errors are a normal part of system operation. At high speeds, periodic bit errors may occasionally be present on the buses and must be handled. These errors are caused by normal clock and voltage noise, along with variations in temperature. While a greater-than-zero bit error rate (BER) is normal and can be expected, higher error frequencies can exist in degrading parts. For instance, if only one data differential pin opens (while the other remains operational), the bus may continue to function but experience a higher error rate (on the order of microseconds or milliseconds).

Lane sparing (data and clock)

As shown in Figure 1, each channel bus interface (upstream and downstream, first and second cascades) has the ability to spare up to two lanes. Solid data lane failures (or those that have a high BER) will be automatically repaired by the TS2–TS7 calibration procedure, either during initial microcode load (IML) or Tier 2 recovery. When the hardware detects a bad lane fault, it steers the logical data to neighboring lanes so that that bad lane is no longer used. Solid clock lane failures will be automatically repaired by the TS0 calibration procedure, either during IML or Tier 3 recovery. Tiered recovery is described later.

Lane-sparing code support

Firmware code keeps track of used and available spare lanes, as well as intermittent spare sample periods and counters per lane. If no spare lanes are available, firmware will make a service request for part replacement. The repair state is preserved over power-on resets; hence, the intermittent failure does not need to be rediscovered. Any spared lane is taken out of commission, before any software has been loaded.

Intermittent data and clock repair

Data errors that fail more slowly than every 25 μs are considered intermittent data errors. These are not automatically self-repaired by the hardware TS2–TS7 training. Instead, syndrome sampling is used to distinguish intermittent single-lane failures from clock or background BER failures. If the syndrome samples do not isolate to a single lane, Tier 3 clock repair recovery is invoked.

Recovery

Advanced memory buffer

The AMB chip is used on several IBM system platforms. It was designed to meet a broad range of RAS needs. We describe the base recovery used on that common buffer chip [9]. For any CRC errors within the AMB subsystem, there is a tiered recovery sequence that is designed to do the following:

1. Detect CRC errors anywhere in the channel, whether on the first or second cascade.
2. Propagate poison CRC back up to the host memory controller to support system-level recovery.
3. Put the DRAMs into a self-timed refresh (STR) state to protect the data.
4. Clear any internal buffers or control states that could have caused a problem.

Once the AMB is in STR mode, other operations can also occur, optionally, as part of the following various tiers:

1. *Tier 1*—Normal resets, retry stores, and fetches across all five channels.
2. *Tier 2*—Recalibrate data interfaces with fast reinitiation. Repair any solid or high-frequency data errors via automatic lane repair. This repair can range from 30 to 100 μs depending on errors and number of cascades. Then, complete Tier 1 cleanup operations and retry stores and fetches across all five channels.
3. *Tier 3*—Recalibrate clock interfaces. Repair any solid clock errors using an alternate clock. This can take approximately 10–20 ms to complete because the clock must be reestablished and the phase-locked loop must be relocked. Then, complete full Tier 2 across the channels, including retrying fetches and stores.

zEnterprise specific

There are several features that are unique to zEnterprise that deviate from the basic channel protocols listed above. These revisions are described in the various recovery tier sections below.

The deviations for recovery for zEnterprise above the recommended lockstep channel recovery are as follows:

1. zEnterprise has line sizes for fetches of 256 bytes. The ECC group for RAIM is 64 bytes (90 bytes of raw data across five RAIM channels). Therefore, four quarter

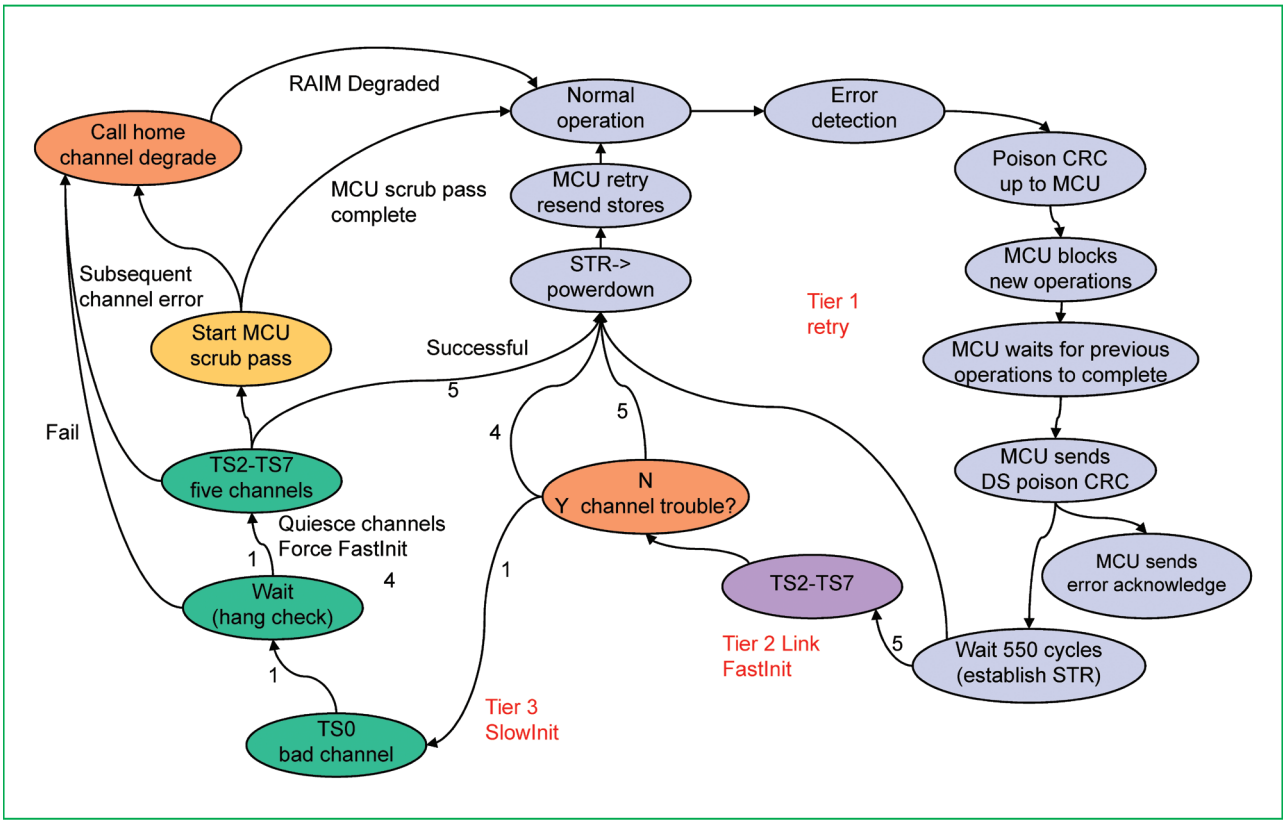

Figure 4

Tier 1, Tier 2, and Tier 3 RAIM recovery. (STR: self-timed refresh; DS: downstream.)

lines are needed to construct a line. No gaps are allowed on the returning data from the memory controller to the L4. Therefore, if the first 64 bytes were sent and a recovery action occurred on a subsequent fetch in the line, there is no opportunity to correct and resend the 64 bytes without causing a gap. Therefore, the ECC needs to correct all the data with no additional latency. This parallel constant latency has been built into the RAIM ECC decoder.

2. If there is a case where an uncorrectable error (UE) is unavoidable, but CRC errors are still present, a special flag is attached to the data. This tag is encoded as a "nonmemory special UE" or a "CRC special UE," which indicates that the UE is from memory, but which a refetch will likely get good data again. This will protect the data from being used but will also lead to a refetch from the core. If the CRC error goes away, the data will be then correctable on the fetch retry.
3. The zEnterprise system is a high-performance server that has an aggressive hang limit from the cores set at 1 ms. Since Tier 3 recovery requires clock calibration and phase-locked-loop locking, the 20–30 ms needed to reestablish the clock would exceed the 1-ms limit. Therefore, zEnterprise uses RAIM "Tier 3 degrade" to allow four channels to run while the bad channel repairs its clock.

Tier 1

Figure 4 shows the control flow for Tier 1 recovery. For any CRC error within a memory channel of the AMB subsystem, the CRC error is first detected, either by the AMB chip or by the MCU. If the AMB detects the error, it immediately blocks all new commands, completes prior commands, and enters the STR state to protect the DRAM data.

At the same time, upstream poison CRC is sent to the memory controller. This signals the MCU that there is a channel problem and recovery must take place. In addition, the bad CRC immediately causes the RAIM ECC decoder to ignore the bad channel using a temporary channel mark that remains active until recovery is completed. Therefore, the fetch data is correctable and is returned to the L4 cache. The MCU will block new operations (fetches and stores) from going out to the channels in order to simplify the recovery process.

Next, the previous (pending) fetch and store operations are completed (within their respective channels). However, the stores remain on a retry queue to be repeated later. The MCU sends poison CRC into all five channels to allow all five channels to reset the same way.

The MCU then sends an error acknowledge to all five channels to reset any interrupts that would cause poison CRC. This is needed to allow the interface to come back later. After a specified amount of time (550 memory cycles), the DRAMs are guaranteed to be in the STR state. Establishment of STR is critical for Tier 1, Tier 2, and Tier 3.

The next step for Tier 1 is to take the memory back out of STR and to enter the power-down state. The memory is now usable. However, since the stores may have been blocked by the AMB chip from writing to DRAMs in the channel that had the initial CRC errors, the previous pending stores are retried.

If there are no new errors on the interface, the new stores and fetches (as well as refresh and periodics) can be sent to the memory and the memory will operate normally again. The Tier 1 recovery takes less than 5 μs.

Tier 2

If there is a hard data or clock error, additional recovery may be required. Forward-progress logic (programmable) monitors whether the mainstream logic is getting processed. When forward progress is not being made, further action is taken. For zEnterprise, Tier 2 is programmable and is invoked after three Tier 1 attempts are made and forward progress is not detected.

During Tier 2, there is not only a quiesce and reset of the channels but there is also a data self-heal step that attempts to repair data lanes that are in error by sparing them out to spare bus lanes. As shown in Figure 4, Tier 2 recovery is identical to Tier 1 recovery except that, after STR is established, the TS2–TS7 link retraining sequence is run on all the channels. This allows for data lane calibration of all data lanes, as well as data lane repair, where needed. The Tier 2 recovery takes from 50 μs to approximately 150 μs depending on conditions.

Tier 3

Once a Tier 2 has been attempted and if there are still problems, the problem channel will be taken offline for clock recovery. The other four channels will continue to run, first exiting recovery, and then performing normal operations. The problem channel will not process fetches and stores; thus, its memory will be refreshed but will be "stale" or old.

Tier 3 will be run on the channel that still has CRC errors (the Tier 3 channel). First, TS0 will recalibrate and repair the clock if needed. Then, if the clock repair/recalibration is successful, the hardware will reestablish STR and will run TS2–TS7 on all five channels. Stores will be retried, and the machine will run normally again. However, there is a Tier 3 channel mark still in place to help correct data from the Tier 3 channel.

The final step that is unique to Tier 3 is "fast scrub." Since the Tier 3 channel may have stale data (from the stores performed to the other channels during Tier 3), the fast scrub step ensures that data from every address is read, corrected (if needed), and stored back with a forced writeback. After that, the Tier 3 mark is removed and the subsystem runs normally again.

In the event that Tier 1, Tier 2, and Tier 3 do not successfully repair clock, data, and intermittent errors and forward progress has not been made, the bad channel is degraded and a service request is issued.

Forward-progress monitor

In order to ensure that Tier 1, 2, and 3 recoveries all work together, it is important to understand whether forward progress is being made while recovery is in progress. For each tier, a programmable window is used to see whether new CRC errors come up within a window. Programmable switches allow particular tiers to run multiple times before attempting recovery of the next tier. If a new CRC error is detected during the forward-progress window and the forward-progress counter exceeds the programmed tier count, the recovery engine escalates to the next tier on the next CRC error.

CRC error/UE trapping

Because the RAIM ECC word is 90 bytes, it is neither practical nor necessary to trap the entire amount of raw data. However, trapping only syndromes causes complications in firmware where these codes have to be decoded. Therefore, in order to better support firmware-driven DRAM and channel-level marking, the trapping hardware in the zEnterprise MCU traps ECC syndromes, as well as DRAM and channel vectors. These vectors identify specific channels and DRAMs that have failed in a given error event within a given rank. Separate traps exist for different types of errors, such as correctable and uncorrectable fetch errors, and per-channel counters exist for bus-level, DRAM-level, and channel-level failures. The trapping stations are programmable, allowing for errors to be considered or not considered for different types of fetch traffic, such as demand fetches versus scrub fetches. Firmware logging provides highly detailed error information by constant monitoring of the error trap and counter hardware.

Maintenance engine

The MCU has several important hardware features that are used to automate most of the memory operations. They are divided into several categories. The maintenance engine within the MCU can perform numerous complex tasks, which can be programmed with variations and called by code atomically as needed. In addition, these tasks can be

dynamically reprogrammed in a running system in order to build even more automation through the firmware and hardware. Most task types also have rate registers, wait registers, or both. Calibration status summary registers are also updated and used. They can be all programmed for IML or for periodic use, as well as to support different DIMM sizes and ranks.

Periodics

Once IML is complete, some DRAM operations need to be started to run periodically. This is all done through programmable list registers. Typically, the periodics will set up a quiesce window and perform all of the valid operations within that window. For instance, the following are some sequences that may show up within a periodic window depending on when the various timers expire:

- Refresh (depends on number of ranks).
- Refresh + ZQ Cal (every 111 ms).
- Refresh + ZQ Cal + Memcal (every 0.89 seconds).
- Refresh + ZQ Cal + Memcal + AMB attention polling (every 3.6 seconds).

Scrub, initiate, test

There is another multipurpose engine called the Scrub, Init, Test (SIT) engine. There are two copies of the engine. Its purpose is to control all the operations that involve one or more addresses in a range, and it is also dynamically programmable via firmware. The following operations are supported:

- Self-test.
- DRAM initialization (clear to zeros during IML).
- Scrub (normal scrub).
- Fast scrub (following a Tier 3 to clean up stale data).
- Test and clear.
- Data test.
- Key test.
- Read/write configuration to the AMB.

Conclusion

The IBM zEnterprise introduces a revolutionary RAIM subsystem design that provides comprehensive RAS protection against myriad failure mechanisms. It prevents server outages from single-channel errors, including bus errors, AMB chip failures, DRAM failures, and calibration problems, and also provides for self-repair of solid and intermittent bus lanes, bus clocks, and DRAM chips while guarding entire DIMMs until replacement can occur. The differential I/O, CRC checking, hardware-assist functions, error polling, and RAIM hardware and firmware all work together to make the memory subsystem robust and highly reliable. The new RAIM design is a standard feature on all zEnterprise servers.

Acknowledgments

Design and implementation of the zEnterprise RAIM subsystem would not have been possible without the combined effort of the entire memory team. The authors would like to thank George Wellwood, Ed Gerchman, Art O'Neill, Katrin Ambroladze, Mike Fee, Pak-kin Mak, Kevin Calabrese, Alia Shah, Bill Wollyung, Fuzi Coe, Matt Pardini, Erica Stuecheli, Dean Bair, Rebecca Gott, Steve Licker, Tom Gilbert, Ed McCain, Victor Fernandes, Tobias Senner, Bryan Cover, Andrew Tsang, Angel Perez, Dorothy Kucar, Nathan Dotson, Dave Rude, Vern Victoria, Arun Nair, Rejo Rekurian, Elmar Gaugler, Elianne A. Bravo, Biagio Pluchino, Kurt Schweiger, Deborah Carbo, Kelly Hopmeier, Rocco Crea, Phyllis Smith, Jim Murray, Dave Cole, Jack Myers, Scott Swaney, Gary Peterson, Ken Christian, Derrin Berger, Rob Reese, Frank Ferraiolo, Dave Cadigan, Jim Keane, Frank LaPietra, Ken Wright, Gary Van Huben, Kevin Gower, Blessy Aleman, Susan Rubow, Doug Search, Mike Wazlowski, Don O'Brien, Ralf Schaufler, Walter Niklaus, Barry Trager, Ashish Jagmohan, Michele Franceschini, and Shmuel Winograd for their contributions.

References

1. B. Curran, J. Warnock, E. Schwarz, L. Eisen, P. Mak, and P. J. Meaney, "zEnterprise 196 system and micro-processor," *IEEE Micro*, vol. 31, no. 2, pp. 26–40, Mar./Apr. 2011.
2. M. Y. Hsiao and C. L. Chen, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," *IBM J. Res. & Dev.*, vol. 28, no. 2, pp. 124–134, Mar. 1984.
3. I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math. (SIAM)*, vol. 8, no. 2, pp. 300–304, Jun. 1960.
4. A. S. Purdy, D. J. Swietek, F. LaPietra, K. C. Gower, and D. J. VanStee, "Memory subsystem technology and design for the z990 eServer," *IBM J. Res. & Dev.*, vol. 48, no. 3/4, pp. 367–381, May 2004.
5. D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1988, vol. 17, no. 3, pp. 109–116.
6. R. Agny, E. DeLano, M. Kumar, M. Nachimuthu, and R. Shiveley, "The Intel® Itanium® Processor 9300 Series: A Technical Overview for IT Decision-Makers," in *White Paper*, Intel, 2010. [Online]. Available: http://download.intel.com/products/processor/itanium/323247.pdf
7. M. Abbott, D. Har, L. Herger, M. Kauffmann, K. Mak, J. Murdock, C. Schulz, T. B. Smith, B. Tremaine, D. Yeh, and L. Wong, "Durable memory RS/6000 system design," in *Proc. 24th Int. Symp. FTCS, Dig. Papers*, 1994, pp. 414–423.
8. F. Busaba, M. A. Blake, B. Curran, M. Fee, C. Jacobi, P.-K. Mak, B. R. Prasky, and C. R. Walters, "IBM zEnterprise 196 microprocessor and cache subsystem," *IBM J. Res. & Dev.*, vol. 56, no. 1/2, Paper 1, pp. 1:1–1:12, Jan./Mar. 2012.
9. G. A. Van Huben, K. D. Lamb, R. B. Tremaine, B. E. Aleman, S. M. Rubow, S. H. Rider, W. E. Maule, and M. E. Wazlowski, "Server-class DDR3 SDRAM memory buffer chip," *IBM J. Res. & Dev.*, vol. 56, no. 1/2, Paper 3, pp. 3:1–3:11, Jan./Mar. 2012.
10. L. A. Lastras-Montaño, P. J. Meaney, E. Stephens, B. M. Trager, J. O'Connor, and L. C. Alves, "A new class of array codes for memory storage," in *Proc. Inf. Theory Appl. Workshop*, 2011, pp. 1–10.

Received January 15, 2011; accepted for publication August 16, 2011

Patrick J. Meaney *IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (meaney@us.ibm.com).* Mr. Meaney received a B.S. degree in electrical and computer engineering from Clarkson University in 1986 and an M.S. degree in computer engineering from Syracuse University in 1991. He is a Senior Technical Staff Member and Master Inventor at IBM as the zSeries* Memory Subsystem Leader. He recently was responsible for designing and delivering the RAIM design on the zEnterprise system. Since joining IBM Poughkeepsie in 1986, he has held cache logic design, timing, and RAS leadership positions on the ES/9021 bipolar-based machines as well as the S/390* zSeries G4, G5, G6, z900, z990, z9*, z10, and z196/zEnterprise complementary metal-oxide semiconductor systems. Mr. Meaney holds 46 U.S. patents and has several patents pending. He has also coauthored several technical papers on RAS. He has received five IBM Outstanding Technical Achievement awards, an IBM Outstanding Innovation award, and an IBM Corporate Award for memory RAIM on the IBM z196 system.

Luis A. Lastras-Montaño *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (lastrasl@us.ibm.com).* Dr. Lastras-Montaño received his B.S. degree in electrical engineering from the Universidad Autonoma de San Luis Potosi and his M.S. (1998) and Ph.D. (2000) degrees in electrical engineering from Cornell University, where he also minored in applied mathematics. Since 2000, he has been a Research Staff Member at the IBM T.J. Watson Research Center, where he is a Master Inventor. Dr. Lastras-Montaño's theoretical research interests are in the general area of information theory, where he has made contributions to the area of point-to-point and multiterminal lossy and lossless compression, large deviations theory, algebraic error control, and modulation coding theory, and more recently a new theory inspired by nonvolatile memories called rewritable channel theory. His practical research interests lie in the area of memory systems, where he focuses on coding for DRAM and emerging memory technologies, as well as novel memory systems architecture and performance analysis. At IBM, he led a team that contributed the technology behind IBM POWER7* error correction code for memory, as well as the POWER7 memory channel reliability features and the compression algorithm behind the POWER7 active memory expansion. He is also responsible for the design of the error control coding algorithm used in the IBM zEnterprise RAIM offering, which enables unparalleled enterprise system availability. In the area of emerging memory technologies, he is leading a team with the goal of making a highly reliable and cost-effective memory technology out of phase-change memory by employing old and new concepts in coding and signal processing. He is author or coauthor of more than 50 journal, conference publications, and issued patents, with more than 60 additional patent applications in process. His work has been recognized on four distinct occasions as an accomplishment of IBM Research and has been given an IBM Outstanding Technical Achievement award and an IBM Corporate Award.

Vesselina K. Papazova *IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (papazova@us.ibm.com).* Mrs. Papazova received an M.S. degree in electrical and computer engineering from Technical University–Sofia, Bulgaria in 1995. She is a Senior Engineer at IBM as the zSeries Memory Subsystem Leader. She was responsible for designing and delivering the memory controller logic on the zEnterprise system. Since joining IBM Poughkeepsie in 2000, she has held design leadership positions on the zSeries z990, z9, z10, and zEnterprise 196 complementary metal-oxide semiconductor systems. Mrs. Papazova holds three U.S. patents and has eight patents pending.

Eldee Stephens *IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (estephen@us.ibm.com).* Mr. Stephens received a B.S.E. degree in electrical engineering from Duke University in 2002. He is a Staff Engineer at IBM responsible for ECC/RAIM hardware in zSeries mainframe systems. He was responsible for designing and implementing significant portions of the memory controller design on the zEnterprise 196 mainframe. Since joining IBM Austin in 2002, he held verification, logic design, physical design, timing, and leadership roles during the POWER6* development effort, as well as served on the hardware bringup teams for the POWER6 processor, the IBM POWER* 570, POWER 575, and POWER 595 systems, and the IBM z10 and zEnterprise 196 mainframe systems. Mr. Stephens has several patents pending.

Judy S. Johnson *IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (jsjohnso@us.ibm.com).* Mrs. Johnson received a B.S. degree from Rensselaer Polytechnic Institute in computer and system engineering in 1983 and an M.S. degree from Syracuse University in computer engineering in 1991. She is a Senior Development Engineer at IBM as the zSeries Millicode Team Leader for memory subsystem support functions. She joined IBM Poughkeepsie in 1983 and the zSeries bipolar machine microcode group in 1991. She is responsible for zSeries bootstrap code and memory subsystem support function code starting with z900 and including the z990, z9, z10, and zEnterprise 196 complementary metal-oxide semiconductor systems. Mrs. Johnson holds U.S. patents and has patents pending. She has received IBM Outstanding Technical Achievement awards and an IBM Outstanding Innovation award.

Luiz C. Alves *IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (alves@us.ibm.com).* Mr. Alves is a Senior Technical Staff Member working in the System z Reliability, Availability, Serviceability Design Group. He graduated from New York University in 1975 with a B.S. degree in electrical engineering and received his M.S. degree in electrical engineering in 1977 from the Polytechnic Institute of New York. He joined IBM in 1977 working in the advanced system manufacturing engineering organization, where he held various technical and managerial positions. In 1985, he was named 3090 Field Quality Assurance Manager, and in 1987, he became the RAS Manager for the 9021 processor families. He is currently responsible for defining the RAS design requirements for future System z products. Mr. Alves has received three IBM Excellence awards, an IBM Outstanding Technical Achievement award, and an IBM Corporate award.

James A. O'Connor *IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (jimoconn@us.ibm.com).* Mr. O'Connor is a Distinguished Engineer at IBM responsible for Systems and Technology Group Advanced System RAS Design and Quality. He joined IBM in 1980 after earning a B.S. degree in electrical and computer engineering from Rutgers University. He later earned an M.S. degree in computer engineering from Syracuse University in 1983 and an engineer's degree (a Ph.D. degree for working professionals) in computer engineering from Syracuse University in 1999. He is a recognized expert in cross-brand common hardware, system quality, and system RAS design and architecture with contributions that span System p*, System z, TotalStorage*, and System x*, and BladeCenter*. He has received IBM Outstanding Technical Achievement awards, an IBM Outstanding Innovation award, and an IBM Corporate Award. He has achieved the sixth patent plateau at IBM and has authored and coauthored several technical papers on RAS system design.

William J. Clarke *IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (wjclarke@us.ibm.com).* Mr. Clarke is a Senior Engineer working in Systems Hardware Development. He graduated from Rutgers University College of Engineering in 1982 with a B.S. degree in electrical engineering. He joined IBM in 1982 as a logic designer working on the 3090 storage subsystem. He joined Product Engineering with the introduction of the 9672 complementary metal-oxide semiconductor processor family. He has coauthored several articles on RAS and is currently the lead RAS engineer for System z196.