Two approaches to array fault tolerance in the IBM Enterprise System/9000 Type 9121 processor

by P. R. Turgeon
A. R. Steel
M. R. Charlebois

The system design of the IBM Enterprise System/9000™ Type 9121 processor was intended to provide high performance and dense packaging within an air-cooled system. Packaging and technology factors had a major influence on the fault-tolerance strategies chosen. This paper describes the effect that this design point had on the fault-tolerant capabilities of two critical 9121 array applications. Although the design challenges faced by these array applications initially appeared to be very similar, the resulting solutions represent very different designs with differing fault-tolerance capabilities. The rationale for these approaches is given, and the error-correction algorithms are described.

Introduction

As discussed by Hajek [1], the design objective of the air-cooled IBM Enterprise System/9000™ (ES/9000™) Type 9121 system was to provide the function and performance of Enterprise System/3090™-class systems, but packaged so as to allow installation outside the water-cooled, raised-floor environments normally required by IBM's high-end general-purpose computing systems. Specifically, the goals were to minimize overall power consumption, to provide cooling via forced air, and to use dense, high-speed logic technologies to achieve the needed function and performance. These engineering constraints led to decisions to use the differential current switch (DCS) circuit family for combinatorial logic and CMOS arrays for dense array applications. Additional factors including cost and configurability led to the selection of 64Kb and 128Kb

CMOS static RAM (SRAM) chips [2] for these memory array applications. These CMOS memory chips were considered dense enough for the proposed applications and operated well within the required power/thermal ranges. However, their access times were less than optimal when viewed against the 9121 performance goals. The specified array access time consumed over 60% of the proposed system cycle time. This had a profound effect on the amount and type of combinatorial logic allowable on any logic paths requiring array access, particularly since the 9121 RAS (reliability, availability, serviceability) objectives mandated the inclusion of array fault-correction and fault-detection algorithms.

Two of the 9121 array applications faced difficult design challenges in meeting the proposed system cycle time and RAS requirements. Although its power consumption and ×32 configuration were attractive from a packaging point of view, the inclusion of the CMOS array chip in the design of the integrated offload processor's (IOP's) local working store (LWS) created the potential for timing problems. The IOP design team therefore adopted a redundant array design which did not sacrifice power consumption and cooling goals in order to meet the cycle time objective and still provide the desired error-correction capabilities. Similarly, the 64Kb CMOS SRAM provided excellent packaging and thermal advantages to the processor cache designers. In this case, however, overall system availability required a more rigorous fault-tolerance capability than the LWS, placing additional design constraints upon the processor cache. The cache design team's solution was to implement a multilevel fault-tolerance approach. The bit-sliced cache partition, combined with byte-level single-bit error correction double-bit error detection (SEC/DED) error-correction code (ECC), provided an excellent fault-detection and -correction capability. The cache line delete (CLD) function then minimized the probability of encountering uncorrectable faults. Thus, what initially appeared to be a similar problem for two critical 9121 arrays evolved into two very different approaches to providing the needed array fault tolerance. This paper discusses these different approaches, examining the rationale for the chosen design directions.

Local working store "shadow"

The 9121 IOP is a modified RISC processor based on the 801 architecture. Its primary functions are to monitor and control 9121 I/O transfers and to participate in I/O subsystem recovery operations. The local working store serves as a scratch-pad memory and as a data buffer for these IOP functions. It is used to hold control block fields containing information which describes the I/O operations currently under way. It is also used to back up critical IOP registers during interrupt handling and error recovery operations.

The local working store is logically organized as an array which is 512 addresses deep by 8 bytes wide. A double word (8 bytes) of data may be written to this array by the IOP during any system cycle. However, because the LWS array write operations are controlled by a mask register, it is possible to write any number of bytes (from 1 to 8) during any array write cycle. Further, these data bytes need not be contiguous. Data are read from the array in a similar manner; on an LWS read operation, 8 bytes of data leave the array. The IOP then selects the actual data byte(s) required.

The desire to provide increased I/O performance capability in the 9121 processor while maintaining a very high level of system data integrity demanded a significantly fault-tolerant LWS design. However, these requirements also placed design constraints on the fault-tolerant techniques which could be used, as follows:

1. Fast data paths. The relatively slow CMOS array chip access time restricted the amount and type of logic allowable in data paths leading to and from the LWS. Simply put, if the 9121 cycle time objective was to be met, logic in the data paths into and out of the LWS had to be kept to an absolute minimum.
2. Noncontiguous byte stores. The ability of the IOP to write any number of bytes under mask register control required fault tolerance at the byte level. Working with wider units of data would have created the need for read-modify-write (RMW) operations, thereby adding unwanted complexity to the logic in the path of data to be written to the LWS. RMW sequences require yet another array access to complete the original write operation, further degrading overall array performance.
3. Erasing correctable faults. To minimize the likelihood that subsequent faults result in uncorrectable errors and to minimize the logic delay penalties for performing corrections, a means to "clean up" soft (not stuck-at or hard) errors should be provided. This also adds complexity to the write paths while requiring additional array accesses.

These discussions led to the abandonment of traditional ECC approaches to fault tolerance for the LWS. SEC/DED ECC across each byte of data provided the necessary fault tolerance but required an amount of additional logic considered unacceptable. While this was not strictly a question of performance, the additional overhead in logic and array chips was detrimental to the overall 9121 design because the TCM upon which the LWS resides was already quite dense and was therefore nearing the maximum allowable power consumption. ECC algorithms across wider units of data, while more efficient, forced a need for read-modify-write operations, as described above. Even b-adjacent types of codes were

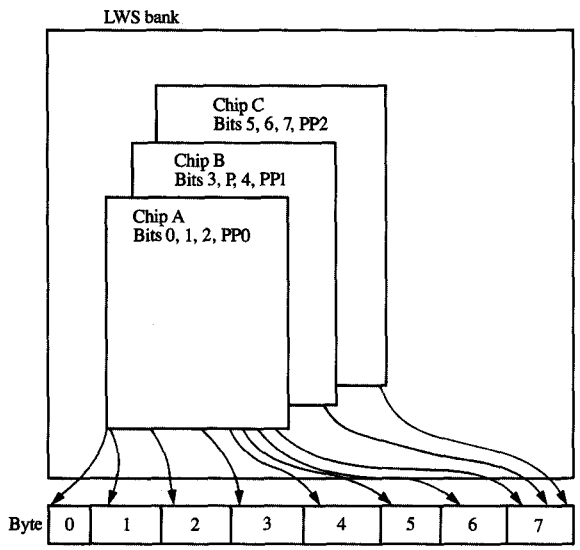

Figure 1

LWS physical layout.

considered and rejected, again because of complexity and potential logic delay in a critical path. It should be noted that none of these approaches directly addressed the need to clean up soft, correctable array faults [3].

For these reasons, an alternative approach involving LWS array redundancy was selected. While redundancy does not intuitively appear to be an optimal solution given the previously stated cost/power objectives, it does allow for a very modular design in the LWS support logic. Although the total chip count (logic + arrays) is no less than for other approaches, the difference is weighted in favor of the considerably lower-power CMOS array chips. The LWS array redundancy therefore played a part in meeting the air-cooling and overall power consumption goals of the 9121 processor. Redundant arrays also eliminated any need for RMW operations, because no byte is dependent upon the contents of any other byte (except its redundant image) for its fault tolerance. Also, the array redundancy minimized the requirements for cleaning up correctable array failures, because good data can reasonably be expected to reside in one half of the LWS array. Finally, this LWS design makes very efficient use of the selected logic technology. As described by Eichelberger [4], the DCS logic cells used in the 9121 processor are based on selector circuits. The LWS shadow design, because it is composed of comparators and selectors, is very efficient in terms of its cell usage and overall performance.

- *LWS shadow design*

The 9121 IOP LWS utilizes six 64Kb CMOS SRAM chips organized into two identical banks of three chips each. One of the LWS banks is called the primary array, while the other is the redundant, or shadow, bank. Data are partitioned within each bank such that each array chip contains three data bits (called a triplet) from each of the eight bytes referenced by an LWS address. Associated with each triplet is a partial parity bit. The data triplets and corresponding partial parity bits are physically partitioned in each LWS bank as depicted in **Figure 1**. The two LWS banks, while unique and separate, share a common data-in bus. This common path helps simplify the LWS write logic. The write mask function is applied at a single point, ensuring that each array receives identical data. From this point on, although the arrays and data paths through the LWS are replicated, the redundancy is totally transparent to the IOP.

Read operations from the LWS array are much simpler and faster than might have been possible with other detection/correction techniques. On an LWS read operation, the read address selects eight bytes from the LWS address space. These eight bytes are read from the primary and shadow banks simultaneously, the data selected from each bank are compared, and the "correct" data are gated onto the LWS data output bus for use by the IOP. Correction, *per se*, is not performed; the correct data bits already resident in one or both of the LWS banks are merely selected onto the active data flow. The LWS shadow may therefore be described as a detection and selection mechanism, rather than a detection and correction algorithm. Thus, the LWS redundancy permits the elimination of the level(s) of logic usually required for data bit correction operations. Also, since the stored data bits are correct in at least one bank of the LWS, there is no need to perform a rewrite of any corrected data back to the array. The LWS shadow design therefore creates a performance advantage by eliminating the need to clean up known soft failures to avoid logic delays to correct bad data.

- *Shadow correction algorithm*

When data are fetched from the LWS, a series of comparisons are performed upon the data leaving the primary and shadow LWS banks before they are routed for use by the IOP. As shown in **Figure 2**, these comparisons are performed in parallel, with the results forming a vector which in turn is used to gate the correct data bits to the data output register.

Three basic types of comparisons are made on the data fetched from the LWS arrays:

1. DD (data-to-data). Each data triplet from the primary LWS bank is compared to the corresponding triplet from the shadow bank.

Table 1 LWS shadow states.

State	*DD*	*PP*	*DP*	*DS*	*Data bank selected*
A0*	0	0	0	0	Primary
B0	1	X	0	1	Primary
B1	1	X	1	0	Shadow
C0	0	1	1	0	Shadow
C1	0	1	0	1	Primary
D0	0	X	1	1	Uncorrectable
	1	X	0	0	
	1	X	1	1	
D1	0	1	0	0	Uncorrectable
	0	0	1	0	
	0	0	0	1	

*No errors.

2. PP (parity-to-parity). The partial parity bits from each bank are also compared.
3. DP/DS. A partial parity bit is generated on each data triplet as it leaves the array chip. This new parity bit is compared to the partial parity bit previously stored. DP compares refer to the check performed on the primary bank's partial parity bits, while DS refers to the comparison performed on the shadow bank's partial parity bits.

The results of this series of comparisons are formed into the vector which describes seven LWS shadow states. These states are summarized in **Table 1**. Briefly, the state vector indicates the condition of the data stored in each bank of the LWS array, and, consequently, which bank should be used as the source for each requested data triplet. For example, when the compare vector indicates that all levels of comparison are successful (no miscompares),

$$A0 = \overline{DD} \cdot \overline{PP} \cdot \overline{DS} \cdot \overline{DP}, \quad (1)$$

the LWS can be said to be in state A0. This state indicates that the data triplet at the selected address is correct in each of the LWS banks. The data from either side of the LWS can safely be used in this case, but by default, data from the primary bank are gated to the data-out register. (This bank designation is programmable. That is, it is possible to define which group of three physical chips is designated as the primary, and which the shadow, via scan-only SRLs. This capability was included for diagnostic and hardware debugging purposes.)

When the triplet comparison (DD) indicates a failure in combination with a miscompare in the generated vs. stored partial parities, one of the LWS banks is known to contain a fault. For example, state B0 is said to occur when the

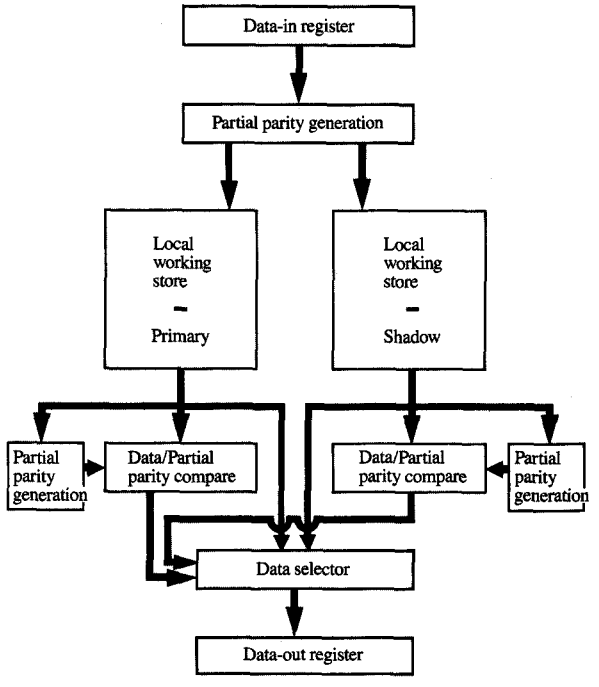

Figure 2
LWS data flow.

DP compare is favorable but the DS compare indicates an error:

$$B0 = DD \cdot DS \cdot \overline{DP}. \quad (2)$$

In this case, the triplet stored in the shadow array is known to contain one or more faults. Therefore, the triplet from the primary bank is gated to the output register. Conversely, when DS compares favorably but DP indicates an error, the primary bank contains the fault. This state (B1) results in the selection of the shadow bank triplet for use by the IOP.

States C0 and C1 occur when the data bits in the primary and shadow banks compare favorably but stored partial parity (PP) and generated vs. stored (DS or DP) comparisons indicate failures. As an example, Equation (3) describes state C1:

$$C1 = \overline{DD} \cdot (PP \cdot DS). \quad (3)$$

States C0 and C1 indicate that one of the stored partial parity bits is incorrect but that the stored data triplets in each bank can be considered to be correct. By default, data from the bank whose stored partial parity was correct

are gated for use. Therefore, data from the LWS shadow are used when state C0 occurs and data from the primary when state C1 occurs.

When the compare vector contains values that suggest multiple, indeterminate faults, state D0 has been entered. In general, state D0 occurs when the results of the three levels of comparison conflict with one another. For example, consider Equation (4):

$$\overline{DD} \cdot DP \cdot DS \cdot \overline{PP}. \quad (4)$$

The DD and PP results imply that all of the stored data and partial parity bits are correct, yet the generated vs. stored partial parity comparisons yield errors. The actual failure may be due to multiple failures in the array chips or to problems in the parity generation logic. In any case, this is an uncorrectable fault. Similarly, state D1 occurs when only a single level of comparison results in a miscompare. Again, the implication is that multiple stored bits are in error or that a control failure has occurred. This, too, is an uncorrectable state.

- *LWS shadow effectiveness*

The 9121 local working store shadow design presented here corrects all single-bit failures within any data triplet. The checking is self-correcting; single failures in the stored partial parity bits are also completely correctable. Furthermore, the LWS shadow design can successfully detect and correct all three-bit failures which fall within a data unit (triplet + its partial parity) and can successfully detect all two- and four-bit failures across this entity. These results have been verified through extensive analysis and simulation.

Processor cache

The 9121 processor cache is a 128KB high-speed buffer designed to match storage access time to the processor cycle time. The cache is the first and fastest memory level in a multilevel storage hierarchy, and it services more than 90% of the storage accesses of the 9121 processor. The cache access time is an order of magnitude faster than the next storage level (main storage). The 9121 processor cache uses a store-in management scheme. In other words, data modified by the processor are held in the cache until needed elsewhere in the system, or until the cache needs to remove them to make room for new data. An alternative approach is the store-through cache, where both cache and main storage are updated whenever data held in the cache are modified. While the store-through design is less complex and offers the advantage of data redundancy, it also increases the usage of the data bus to main memory, an important consideration in systems where multiple processors and I/O servers all vie for access to system memory. The primary benefit of a store-in cache design is therefore to minimize the data traffic to main memory at the expense of more complicated control logic. If the performance and data integrity of the system are to be maintained at their highest levels, the reliability of the store-in processor cache becomes a critical design concern. The cache design team chose a three-faceted strategy to achieve the required high reliability. Briefly, the 9121 processor cache required a high level of fault tolerance, high error-detection capability, and a self-diagnostic and self-maintenance facility to minimize exposure to faults.

- *Cache ECC scheme*

As described earlier, a 64Kb CMOS SRAM was chosen for the processor cache application. It met density requirements, and its organization was suitable for the cache application. From the start, the design goal was to be able to tolerate any single point of failure within any cache array chip. This meant that any array failure, including those which affect the entire chip (chip kill), had to result in a correctable error scenario. A bit-sliced cache design with ECC satisfied this requirement.

The 9121 processor cache uses an ECC scheme which corrects all single-bit errors and detects all double-bit errors within a single byte. This requires five ECC check bits per byte of data. Since the cache is read a double word at a time, 40 check bits are required for each double word of data, a relatively high checking overhead. Although SEC/DED across a double word can be accomplished with a minimum of eight check bits, ECC at the byte level makes feasible a bit-sliced cache design using this CMOS array. Byte ECC also simplifies partial data stores in which fewer than eight bytes have been modified. As in the LWS, the processor cache design eliminates the need for read-modify-write sequences; byte ECC accomplishes this.

The 9121 processor cache is implemented using 26 64Kb chips. The 26 array chips are divided into two identical banks of 13 chips each. This array organization permits two-way interleaving of cached data for better overall system performance. The cache is four-way set-associative; on each read cycle four double words are read from the array. One of these double words is then selected for use by the processor.

The cache array design and layout are depicted in **Figure 3**. The CMOS array chip is used in its ×32 mode; it contains only one data or check bit per byte. For example, consider the array chip corresponding to bit 0. The 32 outputs of this chip are divided into four sets of eight bits to take into account the cache associativity. Within each set, the eight bits represent bit 0 for each byte in the double word. Thus, eight of the thirteen array chips contain data bits 0–7, while the remaining five hold the required check bits. Since no two data or check bits from the same byte reside together in a single chip, any array chip failure within a bank can cause only correctable

errors. Additionally, two chips within the same bank can fail entirely, yet the resulting errors will always be detected.

While the ECC algorithm selected for the 9121 processor cache provides the required ability to detect and correct array faults within the cache, it does add delay to the data read path. To meet cycle time requirements, the path through the CMOS array had to be extended by 3 ns for the processor cache application. This was done by clocking the registers which capture the cache array output with a clock which is 3 ns later than the main system clocks. This results in the loss of 3 ns in the paths fed by these registers, which include the ECC logic paths. Fortunately, the DCS logic set provides for very efficient exclusive-OR trees, the major component of ECC logic. The byte ECC scheme requires only two logic levels for generating syndromes, and two more for syndrome decode and data correction. Thus, the reduced cycle time requirement for the ECC path is met.

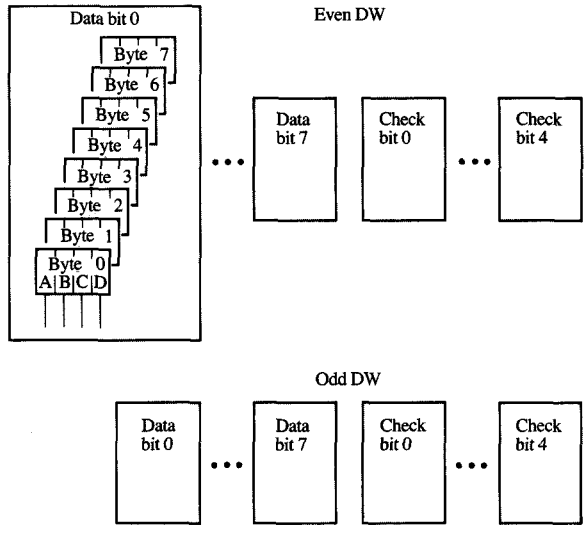

Figure 3

Processor cache layout.

• *Cache line delete*

The processor cache ECC design is capable of correcting all single-bit errors and detecting all double-bit errors. If a cache array chip experiences a hard (stuck-at) failure, the byte containing the error is still usable because the error can be corrected. However, error checking subsequently performed on the line(s) containing correctable faults is essentially reduced to parity checking. Any other failure, whether soft or hard, which aligns itself within an affected byte can only be detected, providing the rationale for the cache line delete (CLD) function. This self-diagnostic feature is designed to allow the system to gracefully detect all hard failures and to minimize any data integrity risks that they might introduce. CLD first appeared in high-end IBM processor caches [5]. It is used to track cache errors and to determine whether they are the result of stuck-at array failures. If a hard failure is detected, the cache line which contains the failing bit is removed from the active cache configuration. This prevents the line from being used and eliminates the potential of another error occurring within the same byte.

The CLD function actually consists of a combination of hardware and licensed internal code (LIC). Whenever a cache access is made, the address and other pertinent information about the access are retained in dedicated registers. If a single-bit error occurs, the bit is corrected, the LIC CLD function is notified that a data correction took place, and the dedicated registers containing information about the corrected array access are read. The LIC checks a CLD table to determine whether this cache line has experienced any other errors within a specified amount of time. If too many errors have occurred in this line during the specified interval, the line is assumed to contain a hard, stuck-at fault and is therefore deleted. Both the time interval and the error-count thresholds are programmable.

Line deletes do not go on indefinitely. The CLD function remains active only long enough to determine that a cache chip is suffering from a fault which affects many addresses. Deleting continues until one of two thresholds has been met. The first concerns the total number of deleted lines allowable within the cache. This value should be high enough to determine that a severe error exists, yet low enough that any loss of performance due to the reduction in cache capacity is negligible. The second threshold involves the total number of deleted lines allowed within a congruence class. A congruence class is defined as the set of all cache lines into which a system address may be mapped. Since the 9121 processor cache is four-way set-associative, there are four lines in each congruence class. At least one line in each congruence class must be available at all times. Therefore, the congruence threshold for this design is three lines or fewer. The deleted line count and congruence thresholds are also programmable.

If a hard failure occurs and it is severe enough to cause a deleted line threshold to be met, the 9121 processor initiates a "call home" via the remote service facility to indicate that the cache has accumulated failures and that a repair action should be scheduled. From this point on, no further lines are deleted. Error correction allows any remaining lines which may be affected by the hard failure

to be read, enabling the system to continue operating normally. Subsequent array failures can cause an uncorrectable error scenario if any byte is affected by two or more faults, but the probability of this occurring during the time between the "call home" and the maintenance action is very small. Thus, the CLD design enhances the overall reliability and availability of the 9121 processor.

- *Cache fault-tolerance effectiveness*

The 9121 processor cache byte ECC algorithm results in the successful detection and correction of all single-bit failures and the detection of all double-bit and many multiple-bit failures. The bit-sliced cache design allows the processor cache to continue functioning even in the presence of a chip kill in either or both cache banks. This design also successfully detects up to two chip kills per bank. The CLD function minimizes the exposure to uncorrectable errors when hard array failures occur. This combination allows the 9121 processor to continue operating normally even in the presence of multiple, severe array failures.

Conclusions

The Enterprise System/9000 9121 design experience confirms that system performance and packaging considerations can and do have a significant impact upon the type of fault-tolerance strategy which can be used. It has also shown that the optimal solution for one type of array application can be very different from the best solution for another, even within the same system.

The 9121 system design objectives led the IOP and processor cache designers to develop solutions which effectively address the issue of fault tolerance for their respective array applications without adverse effect on the overall design goals. Briefly, the local working store shadow approach transformed what was expected to be a serious performance limitation within the IOP into a simple, straightforward design which met all design criteria while efficiently utilizing the technology at hand. The result is a very fast critical array design with excellent detection/correction capabilities. The redundant approach also eliminates correction-related performance penalties. In the case of the processor cache, the integration of the byte ECC algorithm with the cache line delete function results in an extremely reliable cache with excellent self-diagnostic capability. It ensures the integrity of data contained within the cache—often the only valid copy of the data. The fault-tolerance mechanisms employed are unobtrusive and allow full system availability even in the unlikely event of multiple cache array chip faults. The CLD function minimizes the probability of encountering uncorrectable failures by deleting failing cache addresses from the active environment and causing preventive maintenance actions to be scheduled. In short, the LWS shadow and processor cache designs demonstrate that effective and efficient array fault tolerance can be designed despite conflicting system, RAS, packaging, and technology constraints.

Acknowledgments

The authors would like to thank the 9121 IOP and Processor Cache design teams for their assistance in the development of the designs presented here. Special thanks go to Robert Demkowicz and Joe Galioto for LWS shadow implementation, to Bruno Bonetti for the cache ECC algorithm, and to Mike Chan and Dan Elmendorf for the cache implementation. Finally, we would like to express our gratitude to Brian O'Leary and Arnold Tran for their help and guidance with the cache line delete function.

References

1. S. F. Hajek, "The IBM Enterprise System/9000 Type 9121 Air-Cooled Processor," *IBM J. Res. Develop.* **35**, 307–312 (1991, this issue).
2. J. L. Chu, H. R. Torabi, and F. J. Towler, "A 128Kb CMOS Static Random-Access Memory," *IBM J. Res. Develop.* **35**, 321–329 (1991, this issue).
3. C. L. Chen and M. Y. Hsiao, "Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the-Art Review," *IBM J. Res. Develop.* **28**, 124–134 (1984).
4. E. B. Eichelberger and S. E. Bello, "Differential Current Switch—High Performance at Low Power," *IBM J. Res. Develop.* **35**, 313–320 (1991, this issue).
5. B. J. O'Leary and A. J. Sutton, "Dynamic Cache Line Delete," *IBM Tech. Disclosure Bull.*, p. 489 (1989).

Received October 17, 1990

Paul R. Turgeon *IBM Data Systems Division, P.O. Box 950, Poughkeepsie, New York 12602.* Mr. Turgeon is a Development Engineer in the Processor Systems Development organization, currently managing the design and development of future high-end processing systems. Mr. Turgeon graduated from Rensselaer Polytechnic Institute with a B.S.E.E. degree in 1979. That same year he joined IBM Kingston, where he has been involved with the design, development, and RAS aspects of the IBM 8100 Information System, ES/3090 Model S, and ES/9000 9121 processing systems. Mr. Turgeon has received an SPD Division Award (1983, for 8150 processor development) and two DSD Division Awards (1989, for ES/3090 Model S processor development, and 1990, for management of the ES/9000 9121 channel control element design).

Allan R. Steel *IBM Data Systems Division, Neighborhood Road, Kingston, New York 12401.* Mr. Steel is a Staff Engineer in the Processor Systems Development organization, primarily involved with the development of processor caches for large computing systems. He was the lead designer of the ES/9000 9121 processor cache, receiving an Outstanding Technical Achievement Award in 1990 for this effort. Mr. Steel received his B.S. degree in electrical engineering from the New Jersey Institute of Technology, Newark, in 1984, and his M.S. degree in computer engineering from Syracuse University in 1989. He joined IBM in 1984.

Margaret R. Charlebois *IBM Data Systems Division, Neighborhood Road, Kingston, New York 12401.* Ms. Charlebois is a Staff Engineer in the Processor Systems Development organization, currently working on the design and development of high-end processor systems. She received her B.S. degree in electrical engineering from Tufts University in 1984, and her M.S. degree in computer engineering from Syracuse University in 1989. She joined the IBM Data Systems Division in 1984 and has worked primarily in the design and development of the I/O subsystems of IBM mainframe systems. Ms. Charlebois received a DSD Division Award in 1989 for her work in the development of the ES/3090 Model S channel control element. She was the team leader in the development of the ES/9000 9121 integrated off-load processor.