IBM System z9 eFUSE applications and methodology

R. F. Rizzolo
T. G. Foote
J. M. Crafts
D. A. Grosch
T. O. Leung
D. J. Lund
B. L. Mechtly
B. J. Robbins
T. J. Slegel
M. J. Tremblay
G. A. Wiedemeier

IBM System z9™ is the first zSeries® product to use electronic fuses (eFUSEs). The blowing of the fuse does not involve a physical rupture of the fuse element, but rather causes electromigration of the silicide layer, substantially increasing the resistance. The fuse is "blown" with the application of a higher-than-nominal voltage. eFUSEs provide several compelling advantages over the laser fuses they have replaced. The blow process does not risk damage to adjacent devices. eFUSEs can be blown by a logic process instead of a physical laser ablation method. eFUSEs are substantially smaller than laser fuses, and they scale better with process improvements. Finally, since no specialized equipment or separate product flow is required, eFUSEs can be blown at multiple test and application stages. We discuss circuit design, fuse programming, test considerations, and z9™ system applications. The physical and logical implementation of eFUSEs has resulted in improved yield at wafer, module, and final assembly test levels, and has provided additional flexibility in logic function and in system use.

Background

Chip yield, the ratio of acceptable tested chips to the total number tested, is a strong function of the number of memory storage elements, or array cells on chip. Redundant cells have been required on memory chips for many years in order to increase yield [1]. When a defective cell (or collection of cells) was identified through chip testing, the defective address was typically recorded in nonvolatile memory. This was implemented by laser-blown fuses on the chip. Subsequently, whenever the defective address was accessed, the fuses and associated redundancy logic would address redundant cells instead of defective cells. As chip integration has increased, logic has been integrated with large static arrays on most advanced chip designs. Three of five zSeries* chip types integrate logic with memory to such an extent that they require hundreds to thousands of fuses for array redundancy. An electronic chip ID, or EID, has also been implemented with fuses.

Laser fuse (ℓFUSE) methodology, problems, and limitations

ℓFUSEs [2] are blown using a diode-pumped laser. The wafer is stepped to a location specified in a fuse file containing the fuse to be blown (from prior test results) and the physical location. The laser is fired at an empirically determined power and time duration guaranteed to blow the fuse. A chuck steps to the next location, and the process is repeated until all fuses are blown. However, ℓFUSE usage introduces the following problems and limitations:

- *ℓFUSEs use an increasing amount of chip area when redundancy requirements are increasing*. The laser target must be sufficiently large and separated from other structures so that the ablated metal does not cause collateral damage. We have observed that larger chips, advanced semiconductor processes, higher yield, and more stringent test requirements drive the need for more redundancy. This is not just to cover classical "hard" defects, but "soft" defects as well. These soft defects may fail only under certain conditions such as low or high voltage, high speed, or long data retention time.
- *ℓFUSEs do not scale as well as the semiconductor technology*. The target size and pitch depend on the

0018-8646/07/$5.00 © 2007 IBM

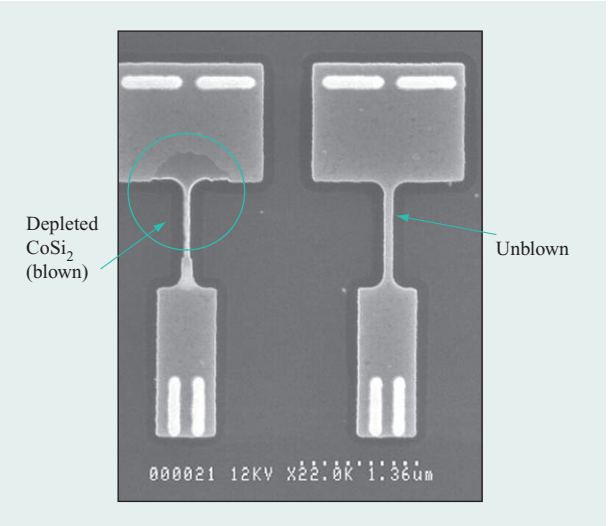

Figure 1

Scanning electron micrograph of blown (programmed) and unblown eFUSEs.

beam spot size, on the need to remain far enough away from other structures to avoid damage, and on stepper accuracy. This indicates an increasing area penalty as semiconductor technology advances.

- *ℓFUSEs require a special laser tool and test flow which lower throughput and restrict ℓFUSE usage*. Fuse blow is typically done after the first pass of wafer testing. After a high-temperature first-pass test, the wafers are routed to the laser tool. After fuses are blown, the wafers are routed back to the wafer tester to verify that the fuses were blown correctly. This second-pass wafer test is done at low temperature. Low-temperature-dependent defects may cause new array fails. Even though these fails may be "repairable," the implementation complexity and additional process time are prohibitive. Chips are processed once. Array fails at any subsequent test step, whether repairable or not, result in the chip being discarded.

These laser fuse disadvantages drove the need for a more flexible and smaller fuse structure.

eFUSE circuit design

The eFUSE [3] implementation used in the System z9* comprises four essential on-chip elements and one essential off-chip element. The silicided polysilicon fuse is the element that is blown or programmed through an electromigration event. **Figure 1** shows a scanning electron microscope image of blown and unblown eFUSEs. The programming circuitry consists of two large-series n-FET transistors designed to draw a large amount of current (10–15 mA), as shown in **Figure 2(a)**. The sense circuitry is the structure that reads the state of the polysilicon fuse, as shown in **Figure 2(b)**. The control logic (not shown) controls the fuse program and fuse read operations. An external voltage source, called F_{source}, is used to program fuse elements (at >3.3 V) and read them (at 0.0 V).

The eFUSE circuit design needs 10–15 mA of current from the F_{source} supply to blow a fuse (F). This required a series n-FET configuration [N0 and N1 in Figure 2(a)] using thick-oxide n-FETs, so that the programming n-FETs were not damaged during application of the high-voltage external supply. The requirement that the programming n-FETs be able to draw 10–15 mA of current through an approximately 200-Ω polysilicon fuse resistor forces these n-FETs to be approximately 50 μm wide. The current requirements during the fuse program and fuse read are significant. This places constraints on the F_{source} wiring in the design. The on-chip wiring for the F_{source} signal was made such that the maximum resistance was less than 5 Ω, and the off-chip wiring was implemented such that the total resistance back to the supply (ground or high-voltage supply) was less than 5 Ω. This requirement was imposed so that the voltage drop on the F_{source} net during a fuse-programming event is less than 100 mV. This guarantees that the external supply is at an accurate voltage, enabling a good fuse-programming event. The other motivation for this wiring requirement is that a large number of fuses (more than 1,000 fuses) must be able to be read at the same time, as is done in the z9* system chips. The risk in reading this many fuses simultaneously is that the voltage divider circuit may change because of a shift of the on-chip ground relative to the ground signal applied on the F_{source} signal. Requiring a low resistance in the F_{source} signal limits this ground shift.

The initial design requirements for the sense circuit [Figure 2(b)] were to interpret any polysilicon fuse of less resistance than 500 Ω as "unprogrammed" and any fuse of greater resistance than 5 kΩ as "programmed." A further requirement for the sense circuitry was that it must not draw more than ~500 μA of current through the fuse to prevent reverse electromigration from occurring. This current limitation sets the size of p-FET P8 in Figure 2(b). This current limit means that in a best-case process and high-voltage condition, the resistance of the parallel combination of p-FETs P8 and P1 must be greater than 2 kΩ. In a worst-case process and low-voltage condition, the resistance of p-FET P8 increases sufficiently that sensing a 5-kΩ fuse as programmed becomes a challenge.

In addition to the above design constraints, the sense circuitry must be protected during application of the

Table 1 State-machine sequence to sense eFUSE.

Time	*Precharge*	*Sense b and precharge*	*Fuse sense 1*	*Fuse sense 2*
0	1	1	0	0
1	0	0	0	0
2	0	0	1	1
3	1	0	1	1
4	1	1	0	0

high-voltage supply, since the oxide breakdown of all FETs in this process is less than 3 V. This requires thick-oxide isolation n-FETs [N1 and N4 in Figure 2(b)] to protect against destruction of the sense circuit during the application of the high programming voltage. To help stabilize the F_{source} voltage during a fuse sense event, an n-FET pulldown [N2, N3, N7 in Figure 2(a)] was added on the F_{source} side of the polysilicon resistor to minimize any "ground bounce" that would occur because of the F_{source} signal.

Since the F_{source} net connects to a chip I/O and then to the outside world, electrostatic discharge (ESD) protection must be provided. Hardware results indicate that the per-fuse n-FET clamp structure of Figure 2(a) is sufficient for ESD protection.

The control circuitry for the eFUSE is fairly simple, consisting of a single latch for each fuse element, tied to the "blow select" input pin, used to select which fuse is to be programmed if the "blow enable" signal is set. The blow enable signal is set by another latch that is controlled independently of the blow select latches. The remaining control logic consists of a set of latches and a small state-machine that goes through the sequence shown in **Table 1** to control the fuse sense.

During this input signal sequence, the following is happening in the sense circuit. The first state is the steady state for the sense circuit, in which it holds the current state in the half-latch structure created by FETs P1, N14, and N15, and inverter I28. The second state is the precharge state, in which the latch feedback loop is broken by turning off n-FET N15 and the node sense node is precharged via p-FET P8. The third state is the fuse sense state, where n-FET N1 is turned on and the voltage divider is set up between p-FETs P8/P1 and the polysilicon fuse through n-FETs N1 and N4. The inverter I28 and p-FET P1 are then used to sense the state of the fuse.

After the initial designs finished long-term reliability testing, a problem with the sense circuitry was discovered. The p-FET P1 [Figure 2(b)] in the half latch used as part of the sense circuitry would weaken over time (because of a voltage-threshold deterioration mechanism known as

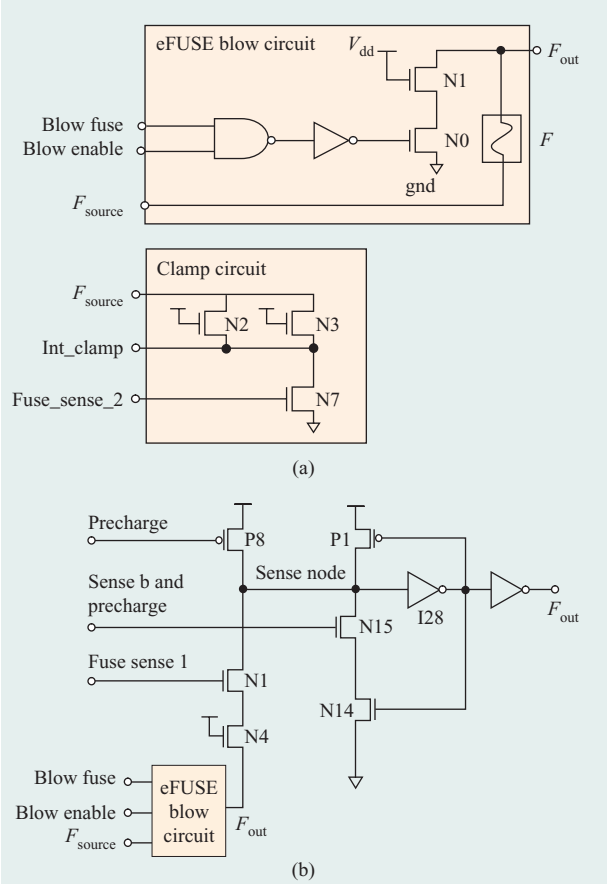

Figure 2

(a) eFUSE programming circuitry; (b) eFUSE sense circuitry.

negative bias temperature instability, or NBTI [4]) if the half latch was storing a 1 value. This happened whenever a programmed fuse was read, and the resulting state would persist until the chip was powered off. The danger is that the degradation of P1 causes a programmed fuse, which can be read correctly immediately after the fuse is programmed, to cease to be read as programmed with the passage of time. This observation on certain chips led to a change in the control logic for the eFUSE sense circuit. This change called for the half latch that is part of the sense circuit to be reset by the control logic so that the P1 does not degrade over time, ensuring that the sense circuit will continue to read an unprogrammed fuse correctly. This change required that a "shadow" latch be added in the control logic. This shadow latch is external to the circuit shown in Figure 2(b), and has its data input tied to the fuse out (F_{out}) pin in Figure 2(b). In addition to the new shadow latch, there is an additional operation performed to clear the half latch. Clearing this half latch is described by the state-machine sequence in **Table 2**.

Table 2 State-machine sequence to clear half latch.

Time	*Precharge*	*Sense b and precharge*	*Fuse sense 1*	*Fuse sense 2*	*Blow fuse*	*Blow enable*
0	1	1	0	0	0	0
1	1	0	1	1	1	1
2	1	1	0	0	1	1
3	1	1	0	0	0	0

During this input signal sequence, the following is happening in the sense circuit. The first state is the steady state for the sense circuit, in which it holds the current state in the half-latch structure created by FETs P1, N14, and N15, and inverter I28. The second state is the precharge state, in which the latch feedback loop is broken by turning off n-FET N15 and the node sense node is precharged via p-FET P8. The third state is the fuse sense state, in which n-FET N1 is turned on and the voltage divider is set up between p-FETs P8/P1 and the polysilicon fuse through n-FETs N1 and N4. The inverter I28 and p-FET P1 are then used to sense the state of the fuse.

eFUSE/ℓFUSE controls and multiplexing (MUX)

There are three basic ways of implementing redundancy through fuse values on System z9 chips, and this flexibility drives the control and multiplexor logic. ℓFUSEs are tried-and-true nonvolatile storage using laser fuses, but they can be programmed only at wafer test. eFUSEs provide potential improved yield and productivity and can be programmed much later in the manufacturing flow (but not in the field). However, they carry the risk of a new type of fuse. "Soft fuses" are implemented by programmable and scannable latches, can be programmed even in the field, and are used to override hard fuses. However, they are volatile storage, and fuse values are lost if power is interrupted. All fuse types were included to minimize schedule risk and maximize yield. The initial chips used ℓFUSE for redundancy because it was a proven methodology on several previous zSeries generations. With a tight schedule, the project could not tolerate any time delays associated with a potential eFUSE problem. eFUSE learning and debug would be done in parallel with functional learning and debug. The final design was to use eFUSE only. Multiplexors determine the fuse data to be used—ℓFUSE, eFUSE, or soft fuse. Soft-fuse latches have a dual use: They may be used to override the other fuses, and they are used to control the programming of the eFUSEs.

Figure 3(a) shows the eFUSE controls, ℓFUSE controls, and multiplexing. The major eFUSE inputs are as follows: The precharge and sense inputs initialize the circuit and read the fuse values into the sense latches. The F_{source} voltage is the power supply used to blow the fuse. The blow control signal is held high when scanning latches and held low to blow a fuse (with F_{source}). The program signal is used to select the fuses to be blown.

eFUSE initialization via the precharge/sense inputs can be done using either test controls or via a state machine, as selected by the eFUSE init select signal. eFUSE initialization via test controls is useful, since it does not require scanning the chip. This is the main method of eFUSE initialization used during manufacturing test. eFUSE initialization via the state machine is used during system operation.

Additional multiplexor controls, fuse select and fuse control, are used to set up the chip to use eFUSEs, ℓFUSEs, or soft-fuse latches as the source for fuse values on the chip (the fuse out signal). Both eFUSE and ℓFUSE values are captured into a non-scan sense latch by an initialization pulse. This value is loaded into the functional soft-fuse latches by the fuse select and fuse control multiplexor signals. Soft-fuse latches are part of the normal scan chain.

Programming (writing)

eFUSE programming is a simple process [see **Figure 3(b)**]. Fuses are blown one at a time in order to limit IR drop on the F_{source} line.

1. Flush zeros into the soft-fuse scan chain (blow control is low).
2. Scan a single "1" into the nth position in the soft-fuse chain, corresponding to the first fuse to be blown.
3. Set blow control high to blow fuse (~200 μs required).
4. Set blow control low, flush chain with zero.
5. Scan in a single "1" to the mth position, corresponding to the second fuse to be blown.
6. Repeat Steps 3 and 4 to blow the next fuse.
7. Repeat Steps 5 and 6, modifying m until all required fuses are blown.
8. Read eFUSEs to verify values.

This "keep it super-simple" approach was chosen to minimize complexity and programming effort. The disadvantage of this approach is that requires many scan cycles to blow all fuses, since the chain always starts with all "0." The EID fuse block, where many fuses are blown to encode the EID, is placed near the beginning of the fuse scan chain to improve programming time. An alternative, more efficient method is to follow Steps 1–3 above but then continue the scan to the next fuse blow position:

4a. Set blow control low but do not flush chain.
5a. Apply *m*–*n* clocks to scan the single "1" to the *m*th position, corresponding to the second fuse to be blown.

After that fuse is blown, the scan is continued to the next position. That fuse is blown. This continues until all fuses are blown. In this way, all fuses are blown with a single scan of the fuse chain. A further enhancement is possible by using multiple F_{source} pins and provides a separate blow control signal for each F_{source} pin.

Reading

Reading is accomplished by generating an initialization signal to read the eFUSE value into its sense latch. The multiplexor controls transfer of the sense latch data to the soft-fuse latch. The steps are as follows:

1. Initializing test controls and state machine.
2. Set eFUSE init select to use either test controls or state machine for precharge/sense eFUSE read sequence.
3. Run precharge/sense sequence to initialize eFUSEs and set eFUSE sense latches.
4. Set fuse select and fuse control inputs to 0 to capture eFUSE values in soft-fuse latches.
5. Scan out soft-fuse latches to verify eFUSE values.

Applications and considerations specific to System z9

eFUSEs were used on 15 of the 16 chips on the System z9 multichip module (MCM). The clock chip used an older semiconductor technology that supported only ℓFUSE. The major applications, referenced in **Figure 4**, are electronic chip ID (EID), array redundancy, recording chip parametric data, implementing fault tolerance on specific arrays, recording MCM specific data, array repair at the MCM level, and recording the total blown-fuse count.

Wafer and package test flow

Electronic fuses can be programmed at each of the following device test steps: high-temperature wafer test, low-temperature wafer test, first-pass package test, and post-stress package test. Wafer test contains two passes, high-temperature and low-temperature. Package test, with the die mounted on a temporary carrier, is done before and after burn-in stress.

The major function of high-temperature wafer test is to remove or repair defective chips. Experience over many years indicates that this test step is most efficient in removing defects, especially those sensitive to low voltage; therefore, this step is usually run first. The EID is a unique code identifying the chip lot, wafer, and *xy* position on the wafer; this is also programmed at the first pass of wafer test. EID eFUSE programming content is identical to that for ℓFUSE. In addition, an intrinsic speed metric using the average performance sort ring oscillator (PSRO) delay is recorded, along with a power sort code based on CMOS quiescent current. This provides the ability to directly read the chip speed and power sort at known temperature and voltage conditions during system diagnostics without referencing an external database. Finally, the count of all fuses blown is programmed into a separate eFUSE bank to serve as a simple checksum.

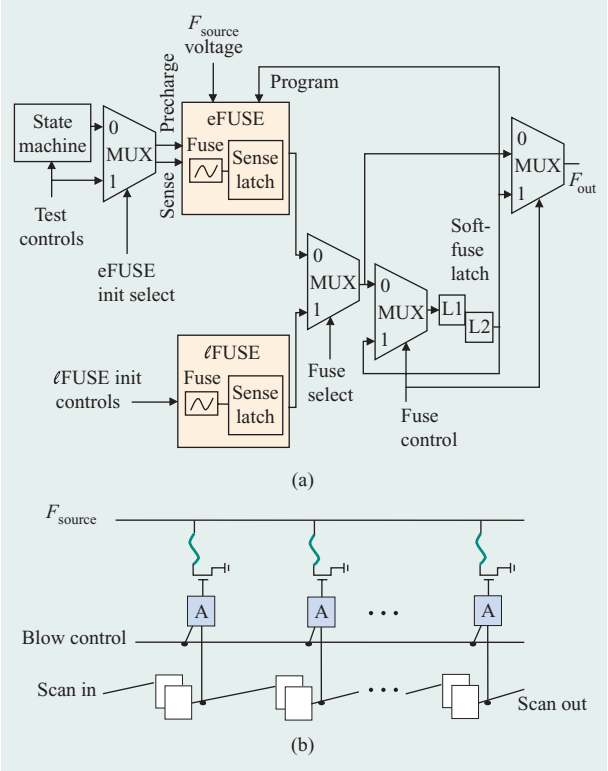

Figure 3

(a) eFUSE/ℓFUSE controls and multiplexing; (b) eFUSE programming logic.

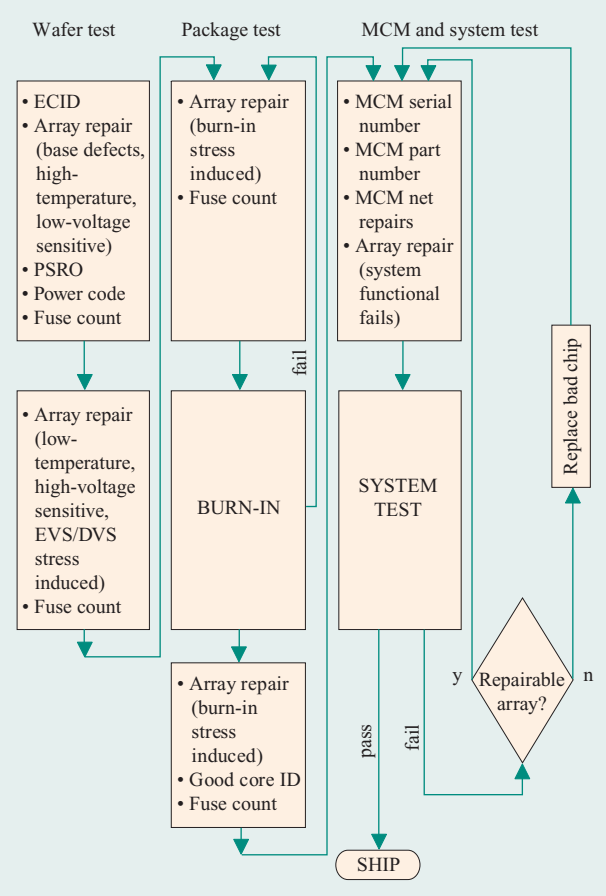

Figure 4

eFUSE applications at various test steps.

Array redundancy eFUSE solutions are determined at all test steps; this results in a significant yield improvement over ℓFUSE. At high-temperature wafer test, a number of array built-in self-test (ABIST) tests are run with different patterns, voltage corners, speeds, and various timing adjustments to check read and write margins. If a chip is marked as possibly repairable, the failed address register (FAR) content is passed to an offline program that merges the fail data from every test to find an optimal solution as the die is being tested. Redundancy fuses are then blown, and ABIST is rerun to verify that the fix worked.

The next wafer test pass is done at low temperatures in order to detect low-temperature-unique fails along with high-voltage-unique fails. Quiescent current is exponentially sensitive to both temperature and V_{dd}. To prevent thermal runaway, many high-voltage tests, including dynamic voltage stress and extended voltage stress (DVS and EVS), must be run at low temperature. At low-temperature testing, the eFUSE repairs are enabled and the ABIST test suite is repeated. If new defects are found, the new FAR contents are passed to the merge program. The merge program in turn checks for available redundancy; if there is any redundancy available, new eFUSEs are blown and the fuse count register is updated. ABIST is rerun, and if the chip fails, it is discarded; otherwise, it is passed on to packaging. The ability to blow fuses at this test step has significantly enhanced yield. New fails caused by EVS or DVS stress could be repaired and recovered, as well as soft fails that occurred only under those conditions. Since these soft fails are also more likely to occur under *in situ* burn-in conditions, burn-in yield is effectively increased.

After dicing, the chip is mounted on a temporary chip attach (TCA) substrate or "package" that provides access to all pins and serves as a carrier for burn-in. Pre-stress package testing is done to check for failures on chip external I/O and defects induced by dicing or substrate attachment. The eFUSE repairs from both passes of wafer test are enabled, and the ABIST test suite is repeated. If new eFUSEs are blown, the count register is updated and ABIST repeated. If sufficient redundancy is not available, the chip is discarded. First-pass fails from the burn-in step are also routinely sent back through this pre-stress test. This permits the recovery of single-cell repairable array fails, one of the most common burn-in fail categories. Our experience showed that approximately 20% of all burn-in fails were recovered by eFUSE array repair.

Since post-stress package testing is similar to prior steps, we discuss it before burn-in. Only passing TCAs from burn-in are sent to post-stress package test. The eFUSE repairs from all prior test steps are enabled, and the ABIST test suite is repeated. If new eFUSEs are blown, the count register is updated to its final value and ABIST is repeated. If sufficient redundancy is not available, the chip is discarded. Multiple chips are then assembled on an MCM and sent to the MCM test sector.

Burn-in considerations

Burn-in, used to improve reliability, is a batch-processing operation in which typically dozens to hundreds of chips from a wafer-processing lot are processed simultaneously. This process involves the application of high-temperature, high-voltage stress conditions while exercising the parts with patterns for a defined duration. IBM zSeries MCM chips use *in situ* burn-in; that is, all tests must be passed under burn-in conditions. Such conditions are well outside the application conditions and can adversely affect eFUSE circuit operation because of NBTI degradation. In addition, it is not necessary for the eFUSE read and write circuitry to operate at stress conditions, since a typical application uses that circuitry only during power-up, a relatively rare event. Tests

were structured to initialize eFUSEs and latch results at nominal conditions only. At stress conditions, only the latched (soft-fuse) information was accessed.

Burn-in fails may be tool-related, batch-process-related, or chip-related. Currents and temperatures are very high. One catastrophic failing chip can cause other chips on the same burn-in board to apparently fail. Finally, the chip itself may fail in either a hard or a soft way. Soft fails are those that occur only at specific conditions. A fast chip may fail only at the burn-in high-voltage and high-temperature conditions and pass at all other conditions. Hard fails are typically reliability fails caused by the acceleration of a defect mechanism. A hard fail is a part that originally passed but now fails at non-stress conditions. Because of these various failure modes, all chips have two chances to pass burn-in. So-called first-pass burn-in fails are all collected in a new "re-con" lot and sent back to pre-stress package testing. Passing parts continue to second-pass burn-in, and second-pass burn-in fails are discarded.

The burn-in system has the capability to test and extract data from individual parts, but it cannot apply a unique repair action to each part. Software and a common database structure (see **Figure 5**) were designed to capture array fail information from the FAR for a given ABIST test, the burn-in lot, and the chip EID under burn-in conditions. Multiple ABIST tests were run, and the FAR data for each was fed to a merge program to calculate the optimum repair. This merged burn-in repair data, along with lot and EID, was stored in the database. As previously discussed, all first-pass burn-in fails were sent back to pre-stress package test. That tester had access to the database containing the burn-in repair information. It was planned to use the burn-in repair data to blow eFUSEs to repair *in situ* burn-in fails. In practice this was not needed, because the burn-in yield exceeded its target. Stress-induced repairable fails (for example, new single-cell array fails at non-burn-in conditions) are a major fail mechanism captured by this methodology. Since these fails are sent back to pre-stress package test, that step will identify the new fail.

eFUSE at MCM test

A major eFUSE advantage is the ability to program at the multichip-module (MCM) level. MCM F_{source} wiring was designed specifically to provide the ability to program eFUSEs on all MCM chips. There were two main MCM eFUSE applications in manufacturing: MCM electronic ID (EID) and MCM yield recovery.

Every MCM has a unique EID programmed in eFUSEs that contains the MCM serial number and the MCM part number. The EID can be retrieved electronically at any packaging level that uses this MCM. Information such as MCM build history, chips on MCM

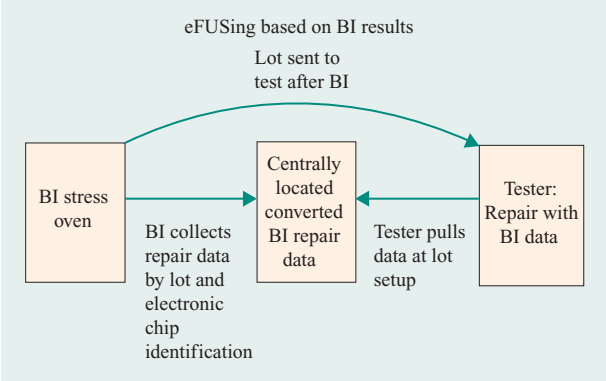

Figure 5

eFUSE burn-in (BI) dataflow.

vintage, and MCM quality level can be retrieved. Critical test information is saved by EID for future reference. This information has been proven useful for comparing MCM characterization data among wafers, chips, and systems. It has also been used extensively for MCM traceability. Two separate z9 system eFUSE applications improve MCM yield: eBIT[1] implementation to recover substrates with defective nets, and functional array repair on the large cache chip (called SCD) without chip replacement.

The design of the MCM and chip I/O provides redundant net capability to maximize substrate yield [5]. There are approximately 380 repairable groups of nets, with roughly one spare net for every 12 functional nets. In manufacturing, all functional nets are tested for opens with an interconnect test [6]. All substrates have a database entry indicating the defective (open) nets on that substrate. If all fails from the interconnect test match all entries in the open substrate net list, codes uniquely indicating every failing group and net are burned into eBIT eFUSEs on the memory controller chip called MSC0. This chip was chosen because it had many spare eFUSEs. If the interconnect test results do not match the substrate fail list, the MCM is sent for physical repair, which usually results in the removal of two chips. System-level code accesses the eBIT fuses and reconfigures logic to use the redundant net in place of the open net. The estimated quantity of substrates that can be saved by eBIT is about 5%.

Another method of enhancing MCM yield and reducing rework time and cost is to repair functional array fails at MCM test without chip replacement. ABIST is very effective at catching array fails. However, it has been our experience over multiple programs that

[1]eBIT: General term used to identify all allowable defective nets that can be repaired (spared) electronically by eFUSEs.

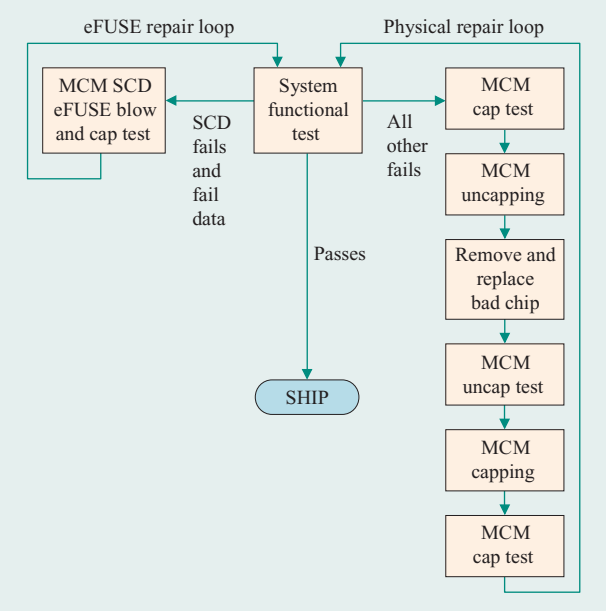

Figure 6

eFUSE and physical MCM repair loops.

some array fails escape to final assembly test, and these are consistently one of the most common causes of MCM rework. The four SCD cache chips on the MCM contain in total more than 400 million array cells. The chip also has the capability to catch the failing address register (FAR) in many functional fail cases. The FAR contents indicate whether the fail is repairable. There is no capability at system level to blow eFUSEs, but there is at the MCM test level. Referring to **Figure 6**, code was written to capture the MCM EID, chip EID, and FAR at final assembly (system) test. This data is transmitted to the MCM test sector. This failing MCM does not go through the normal physical repair loop, but instead through a much shorter path. eFUSEs are blown at MCM test on the chip identified by functional system test. The MCM is retested to ensure that nothing else is defective, and it is then sent back to system test. The time and expense of uncapping, physical chip replacement, uncap test, and recapping is avoided. Approximately 10–12% of System z9 MCM line returns were repaired by eFUSEs.

Array fault tolerance test considerations

The processor chip used multiple copies of a relatively small array called a content-addressable memory (CAM) that caused significant yield problems on a predecessor chip. These problems were due to localized single-cell fails in the array. Because the CAM array design contained no redundancy, a defect in any one of the instances of this array would cause the loss of at least one of the two cores, or possibly the entire chip. Layout changes were made to the CAM, but it was unclear whether they would fix the problem. The addition of redundancy was not possible. It would be a major change to the array, and it would have a major impact on schedule and chip layout; another solution was needed.

Each CAM had four separate quadrants, and it was decided that the system could accept a single defective quadrant. Functional fault tolerance on CAM was already in place as part of the z9 system reliability, availability, and serviceability (RAS) strategy, but this was meant to quarantine a defective quadrant in the rare event of a reliability fail so that functional operation would continue. This is very different from applying structural logic and array built-in self-test (LBIST and ABIST) tests to a chip in which any one quadrant in any CAM could be bad but the chip would still be good. It was theoretically possible that every processor core on the MCM could have a fail in a different quadrant, yet the same tests could be used for all chips. A novel fault-tolerant structure using eFUSEs was chosen to help improve yield should this CAM problem reoccur.

Each CAM quadrant has a dedicated multiple-input signal register (MISR) [7], in which quadrant responses to the stimulus supplied by BIST testing are compressed. A defective quadrant affects the signature only in its dedicated MISR. Typical single-cell defects do not affect LBIST, since LBIST uses only the write-through array function and does not read array cells; ABIST, of course, is affected. The test-and-tag procedure was as follows **(Figure 7)**. Initially, eFUSEs to tag the bad CAM quadrant are not set. If the ABIST test program detects a bad CAM MISR signature, an eFUSE is blown, indicating the specific bad CAM and quadrant. If ABIST is repeated, logic inside the array senses the set eFUSE and forces the quadrant data-out pins and the MISR to a known constant value. There are four independent CAMs per core, each with four quadrants, giving sixteen possibilities for acceptable defect locations; with the one nondefective case, that gives seventeen possible "good" MISR signatures per core. The test program reads the CAM eFUSEs, and on the basis of what is set, selects one of the seventeen ABIST MISR signatures as valid. If the MISR matches that signature, the CAMs are called good. A CAM fuse could be set at any test step from wafer through package post-stress, but only one quadrant defect would be allowed.

A more comprehensive type of LBIST, known as LBIST_COMBINED, will, on the other hand, write and read array cells. Theoretically it could be treated similarly to ABIST: Choose one of seventeen LBIST_COMBINED signatures based on the CAM eFUSE that was set. In practice, this was more

complicated because of the number of possible combinations. LBIST_COMBINED is a suite of tests, each with different clocking, pattern counts, and weighting. Each separate pattern would require seventeen different valid signatures. Fortunately, the layout change fixed the CAM yield problem; none of these options, including ABIST, were needed.

System-level implementation

EID information for each chip is read by the System z9 support element (SE) code from eFUSEs and stored in several files on the SE hard disk—for example, the self-test MISR results file and a general file that records all eFUSE values. EID information is kept and logged by the SE to allow for automatic parts tracking in manufacturing.

The MCM serial number and MCM part number are also available in eFUSEs for the SE to read and store. SE code gathers this information as well as other chip-specific information (such as sort, PSRO, and flush delay from eFUSEs on MSC0) and saves it to a system data file tied by serial number to a specific MCM. This system summary file is used to track chip performance during characterization and in analysis of chip fails in manufacturing.

Known array failures on every chip are burned into the eFUSEs on that chip at manufacturing. The SE reads these blown eFUSEs and stores them in a table for use at IML time to repair these arrays by using redundant cells. The array repairs are controlled by the SE by transferring the table contents to the soft-fuse latches using the fuse control bits. These array repairs are applied during a system initialization procedure after power-up called initial machine load (IML). The array repairs are preserved during subsequent IMLs. This process is substantially the same as the process previously used with laser fuses.

Chip interface failures discovered in manufacturing are also marked for repair with the use of eFUSEs. Up to eight repairs are possible using redundant chip interface wires. The SE reads these repairs from the eFUSEs into a table stored on the SE hard disk and applies them during IML by scanning appropriate latches. This allows chip interface fails to be repaired instead of scrapping the MCM.

The virtual power-on process [8] was used to verify this SE code prior to system power-on. The eFUSEs were "blown" by updating with the encoded data in the repair scan chain in the simulation model. Subsequently the steps in the repair process, in which SE code reads the blown eFUSE data and builds the repair table to be applied later in IML, were verified before real hardware was delivered.

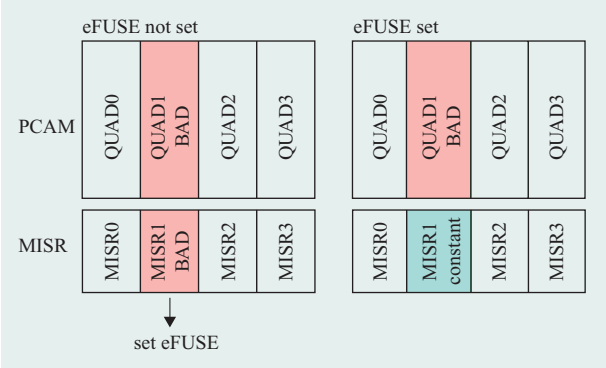

Figure 7

CAM fault tolerance.

Conclusions

Improved yields at all test levels are directly attributable to eFUSEs. They are easily programmed, easily read, and reliable. New features that were previously not practical are now feasible; these include burn-in yield recovery, multichip module (MCM) electronic ID, MCM substrate repair, and MCM array repair without chip replacement. Since eFUSEs are significantly smaller than laser fuses and scale with the technology, more redundancy is available for less area. It is extremely convenient and elegant when pertinent test and redundancy data resides on the chip or MCM itself, with no external lookup required. The main challenges to eFUSE technology are from database lookup methods or electrically erasable programmable read-only memory (EEPROM) implementations, in which redundancy information is no longer stored on the chip as it is needed. This is driven by a predicted steep increase in required redundancy based on more dense arrays, process variation, projected defect levels at volume ramp time, and so on. Cost is also a factor. Real estate on leading-edge ten-level-metal, 60-nm silicon-on-insulator (SOI) process chips is much more expensive than on an EEPROM chip. Compression schemes have been proposed [8] to reduce the number of fuses and the associated area required. It remains to be seen whether the complexity and time associated with database lookup, availability, and maintenance outweighs its obvious area advantage. There is an attractive simplicity in the "plug-and-play" aspect of eFUSE technology, with no external referencing required, especially for critical elements such as array redundancy.

Acknowledgments

Many individuals contributed to the success of IBM System z9 eFUSE development. The authors particularly thank the following: In East Fishkill, New York,

Franco Motika, Donato Forlenza, Orazio Forlenza, William Hurley, and John Parker from MD test and characterization, and Subu Iyer, Alan Leslie, and Chandrasekara Kothandaraman in the eFUSE development group; in Boeblingen, Germany, Otto Torreiter for characterization and debug; in Poughkeepsie, New York, Rick Dennis and Chris Berry for physical design, Steve Michnowski, Ron Frishmuth, and Humayun Kabir for test pattern development, Tom Gilbert for simulation, Ed McCain for system verification, Steve Wilson for test, and Phil Wu and Dan Skooglund for management support; in Essex Junction, Vermont, James Pettine and Gary Sarnowicz for test development, Mark Ollive for merge program development, Greg Miller for burn-in development, Tami Vogel and Keith Stevens for physical failure analysis, Janet Rocque and Deb Hamm for logistical support, and Matt Ringler for eFUSE development and initial design.

References

1. P. Bunce, J. Davis, T. Knips, and D. Plass, "System for Implementing a Column Redundancy Scheme for Arrays with Controls that Span Multiple Data Bits," U.S. Patent 6,584,023, June 24, 2003.
2. T. H. Daubenspeck, T. L. McDevitt, W. T. Motsiff, and A. K. Stamper, "Triple Damascene Fuse," U.S. Patent 6,667,533, December 2003.
3. C. Kothandaraman, S. K. Iyer, and S. S. Iyer, "Electrically Programmable Fuse (eFUSE) Using Electromigration in Silicides," *IEEE Electron Device Lett.* **23**, No. 9, 523–525 (2002).
4. Y. Lee, S. Jacobs, S. Stader, N. Mielke, and R. Nachman, "The Impact of PMOST Bias-Temperature Degradation on Logic Circuit Reliability Performance," *Microelectron. Reliabil.* **45**, 107–114 (2005).
5. D. M. Berger, J. Y. Chen, F. D. Ferraiolo, J. A. Magee, and G. A. Van Huben, "High-Speed Source-Synchronous Interface for the IBM System z9 Processor," *IBM J. Res. & Dev.* **51**, No. 1/2, 53–64 (2007, this issue).
6. O. Torreiter, U. Baur, G. Geocke, and K. Melocco, "Testing the Enterprise IBM System/390* Multi Processor," *Proceedings of the IEEE International Test Conference*, 1997, pp. 115–123.
7. T. Foote, D. Hoffman, W. Huott, T. Koprowski, B. Robbins, and M. Kusko, "Testing the 400MHz IBM Generation-4 CMOS Chip," *Proceedings of the IEEE International Test Conference*, 1997, pp. 106–114.
8. M. R. Ouellette, D. L. Anand, and P. Jakobsen, "Shared Fuse Macro for Multiple Embedded Memory Devices with Redundancy Compression Scheme," *Proceedings of the Custom Integrated Circuits Conference*, 2001, pp. 191–194.

Received May 5, 2006; accepted for publication October 17, 2006; Internet publication February 13, 2007

Richard F. Rizzolo *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (rizzolo@us.ibm.com).* Mr. Rizzolo is a Senior Technical Staff Member and served as test team leader for several System z* projects, most recently for the System z9. He also has primary responsibility for the sort and characterization methodology for MCM chips designed in Poughkeepsie, New York. He received his B.S. degree in physics from Rensselaer Polytechnic Institute in 1977 and his M.E. degree in electrical engineering from Rensselaer in 1980. Since joining IBM in 1978, Mr. Rizzolo has worked on bipolar and CMOS projects in the areas of design for testability, high-frequency design and timing analysis, diagnostics, and circuit design. He holds ten patents and has co-authored a number of papers in the field of testability and diagnostics.

Thomas G. Foote *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (tomfoote@us.ibm.com).* Mr. Foote is a Senior Engineer and test data development team leader for zSeries chips. He received his B.S.E.E. degree from Purdue University in 1973, joining IBM that same year in the area of chip test data development for early FET chip designs. Mr. Foote has worked in test tool development, functional simulation, packaging tools, large systems competitive analysis, and most recently in chip test data development. He holds patents and has co-authored papers in the area of design and design for test.

James M. Crafts *IBM Systems and Technology Group, 1000 River Street, Essex Junction, Vermont 05452 (jcrafts@us.ibm.com).* Mr. Crafts is a Senior Engineer in the World Wide Test Engineering Group. He joined IBM in 1985.

David A. Grosch *IBM Systems and Technology Group, 1000 River Street, Essex Junction, Vermont 05452 (grosch@us.ibm.com).* Mr. Grosch received a B.S. degree in electrical engineering from the Rochester Institute of Technology in 1985, joining IBM that same year, He is currently an Advisory Engineer working on zSeries burn-in development.

Tak O. Leung *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (tleung@us.ibm.com).* Mr. Leung is a Senior Engineer and the MCM manufacturing test leader for zSeries MCMs. He received his B.S.E.E. degree from Columbia University in 1983, joining IBM that same year in the area of chip design-for-test for vendor components and continuing to work in the field of test development for both large-system MCM designs and vendor MCM designs, with a focus in manufacturing. Most recently Mr. Leung held program management responsibilities for the MCM manufacturing of the bond, assembly, and test operations for several OEM customers. He has co-authored papers in the area of design and design for test.

David J. Lund *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (dlund@us.ibm.com).* Mr. Lund received an M.S. degree in computer science from Union College in 1985. In 1987, he joined IBM at the Poughkeepsie, New York, Development Laboratory to work on the System/390* Processor Controller. He is currently a Senior Software Engineer, working on the zSeries service element. He has received several IBM Outstanding Technical Achievement Awards for his work on System/390 and zSeries hardware reset applications.

Bryan L. Mechtly *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (mechtly@us.ibm.com)*. Mr. Mechtly received a B.S. degree in computer science from Indiana University of Pennsylvania in 1983. After graduation he joined IBM in the Processor Controller Department in Poughkeepsie, New York, working on bipolar mainframe computers. Mr. Mechtly supported the transition to CMOS mainframe technology and currently works in the service element area. He has received several IBM Outstanding Technical Achievement Awards for his work on the processor controller and service element; he is a coauthor of one U.S. patent and several patent applications.

Bryan J. Robbins *IBM Systems and Technology Group, 3790 Mesquite Drive, Beavercreek, Ohio 45440 (brobbins@us.ibm.com)*. Mr. Robbins is a Senior Engineer at IBM. He received a B.S. degree in electrical engineering in 1986 and an M.S. degree in computer and electrical engineering in 1988, both from Purdue University. Mr. Robbins has co-authored several papers in the area of design for test; he holds eight patents.

Timothy J. Slegel *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (slegel@us.ibm.com)*. Mr. Slegel received his B.S.E.E. and M.S.E.E. degrees from Lehigh University in 1980 and 1982, respectively, joining IBM in 1982. He has worked in many areas of processor design, including floating-point units, vector processors, and cache design. He was the chief architect and overall team leader for the G5 and z990 microprocessors, and the overall technical leader for the processor subsystem in the System z9. Mr. Slegel has received two IBM Corporate Awards, three IBM Outstanding Innovation Awards, two IBM Outstanding Technical Achievement Awards, and a Tenth-Plateau IBM Invention Achievement Award, with 38 U.S. patents. He is a Distinguished Engineer, currently working on the design of future IBM systems.

Michael J. Tremblay *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (MJT307@us.ibm.com)*. Mr. Tremblay received B.S. and M.S. degrees in electrical engineering from the University of New Hampshire. He also received an M.B.A. degree in finance and accounting from Marist College. He was employed as a Senior Engineer by G.T.E. before joining IBM at the Poughkeepsie, New York, Development Laboratory to work on the 3080X System Processor Controller. Mr. Tremblay was a key test team member for the H2 MCMs, the first modules to incorporate self-test. He was the system architect for the SST MCM testers designed and built by IBM. Mr. Tremblay has received an IBM Outstanding Technical Achievement Award for his work on the H2 MCM self-test design. He has presented papers at the Manufacturing Technology Symposium and the Test ITL.

Glen A. Wiedemeier *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (wiedem@us.ibm.com)*. Mr. Wiedemeier is a Staff Engineer; he has served as an I/O designer for POWER4+*, POWER5*, POWER5+*, and z9 systems. He received his B.S. degree in engineering from the University of Wisconsin at Madison in 2001. He joined IBM that same year and has since designed high-speed single-ended and differential amplifiers, laser fuse sense circuits, and eFUSE sense circuits.