# Intel StrataFlash$^{TM}$ Memory Technology Development and Implementation

Al Fazio, Flash Technology Development and Manufacturing, Santa Clara, CA. Intel Corp.
Mark Bauer, Memory Components Division, Folsom, CA. Intel Corp.

Index words: StrataFlash, MLC, flash, memory.

## Abstract

This paper will review the device physics governing the operation of the industry standard ETOX$^{TM}$ flash memory cell and show how it is ideally suited for multiple bit per cell storage, through its storage of electrons on an electrically isolated floating gate and through its direct access to the memory cell. The device and reliability physics aspects of the three key technology features of multiple-levels-per-cell (M.L.C.): precise charge placement, precise charge sensing, and precise charge retention are discussed. The mixed signal design implementation of these features is reviewed along with challenges for low periphery circuit overhead and standard flash memory product performance. Lastly, process manufacturing aspects are reviewed and it is shown how Intel StrataFlash$^{TM}$ memory is manufactured on the same process flow and at the same high yields as standard flash memory.

## Introduction

The concept of M.L.C. is ideally suited to the flash memory cell. The cell operation is governed by electron charge storage on an electrically isolated floating gate. The amount of charge stored modulates the flash cell's transistor characteristic. M.L.C. requires three basic elements: (1) Accurate control of the amount of charge stored, or placed, on the floating gate such that multiple charge levels, or multiple bits, can be stored within each cell, an operation called placement; (2) accurate measurement of the transistor characteristics to determine which charge level, or data bit, is stored, an operation called sensing; and (3) accurate charge storage, such that the charge level, or data bit, remains intact over time, an operation called retention. These elements are achieved by exploiting stable device operation regions and by the direct cell access of the ETOX flash memory array.

## Flash Cell Structure and Operation

An explanation of M.L.C. first requires a review of the flash memory cell. The ETOX flash memory cell and products[1] have a long manufacturing history, having evolved in the late 1980's from EPROMs, which had been an industry standard from the early 1970's.

### Cell Structure

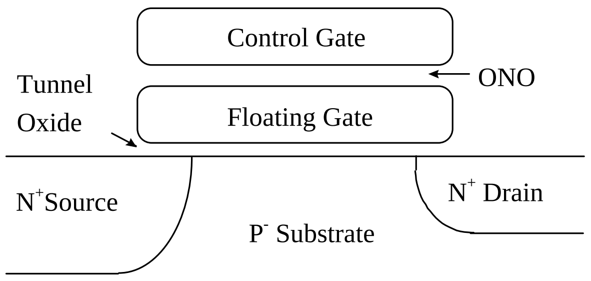

**Figure 1: ETOX flash memory cell cross section**

Figure 1 shows a cross-sectional view of a flash cell. It consists of an N-channel transistor with the addition of an electrically isolated poly-silicon floating gate. Electrical access to the floating gate is only through a capacitor network of surrounding $SiO_2$ layers and source, drain, transistor channel, and poly-silicon control gate terminals. Any charge present on the floating gate is retained due to the inherent Si-$SiO_2$ energy barrier height, leading to the non-volatile nature of the memory cell. Characteristic of the structure is a thin tunneling oxide (~100**Å**), an abrupt drain junction, a graded source junction, ONO (oxide-nitride-oxide) inter-poly oxide, and a short electrical channel length (~0.3µ). Because the only electrical connection to the floating gate is through capacitors, the flash cell can be thought of as a linear capacitor network with an N-channel transistor attached. The total capacitance of the cell ($C_{TOT}$) is equal to the additive capacitance of the network. For convenience, coupling ratio terms, which are defined as the ratio of terminal voltage coupled to the floating gate, can be defined as follows:

GCR =control gate coupling ratio,

DCR = drain coupling ratio, and

SCR = source coupling ratio.

Therefore, a change in control gate voltage will result in a change in the floating gate voltage, $\Delta V_{FG}=\Delta V_{CG}*GCR$. The basic equation for the capacitor network is

$$V_{FG}=Q_{FG}/C_{TOT} + GCR*V_{CG} + SCR*V_{SRC}+DCR*V_{DRN} \quad (1)$$

where $Q_{FG}$ = the charge stored on the floating gate.

A simple first-order transistor equation of drain current says

$$I_D=G_M*(V_{FG}-V_{CG}-V_{DRN}/2)*V_{DRN} \quad (2)$$

where $G_M=q\mu eC_{OX}Z_E/L_E$

This equation is very inexact for the small geometry of the flash cell, but nevertheless the conclusions hold. Substituting $V_{FG}$ of the basic coupling ratio Equation (1) into the basic transistor I-V Equation (2) leads to the conclusions that the transconductance of the transistor (and also the pre-threshold slope) degrades by GCR, while the threshold voltage, $V_T$, depends upon $Q_{FG}$, the charge stored on the floating gate. Therefore, the $V_T$ depends upon $Q_{FG}$, while the I-V shape does not. Very simply, the flash cell can be thought of as a capacitor which is charged and discharged, the charge value being determined by the amplification of the transistor I-V. To give an idea of the amount of charge, every volt of cell threshold corresponds to approximately 10,000 electrons of floating gate charge.

**Cell Operation: Programming**

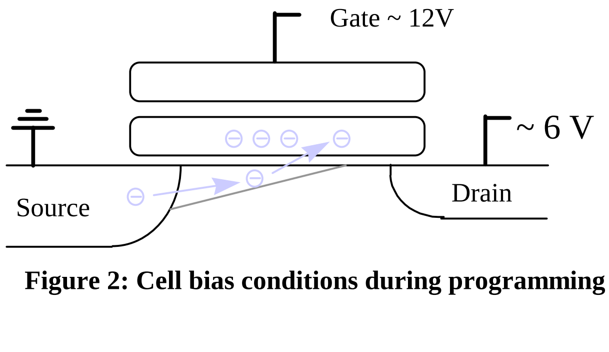

**Figure 2: Cell bias conditions during programming**

Programming a flash cell means that charge, or electrons, are added to the floating gate. Figure 2 shows the cell bias conditions during program operation. A high drain to source bias voltage is applied, along with a high control gate voltage. The gate voltage inverts the channel, while the drain bias accelerates electrons towards the drain. Programming a flash cell, by channel hot electrons, can be understood by use of the lucky electron model[2], as illustrated by the energy band diagram in Figure 3. In the lucky electron model, an electron crosses the channel without collision thereby gaining 5.5-6.0eV of kinetic energy, more than sufficient to surmount the 3.2eV Si-$SiO_2$ energy barrier. However, the electron is traveling in the wrong direction. Its momentum is directed towards the drain. Prior to entering the drain and being swept away, this lucky electron experiences a collision with the silicon lattice and is re-directed towards the Si-$SiO_2$ interface, with the aid of the gate field. It has sufficient energy to surmount the barrier. However, an electron does not have to be completely lucky. It can be "somewhat lucky" or "barely lucky," making the process of programming efficient. We can observe from this model that the lateral field, determined by bias voltage, junction profiles, electrical channel length, and channel doping are important to the effectiveness of generating energetic electrons and are therefore key to the M.L.C. placement operation. Hence the abrupt drain junction and short channel length of the cell structure. After programming is completed, electrons are added to the floating gate, increasing the cell's threshold voltage. Programming is a selective operation, uniquely occurring on each individual cell.

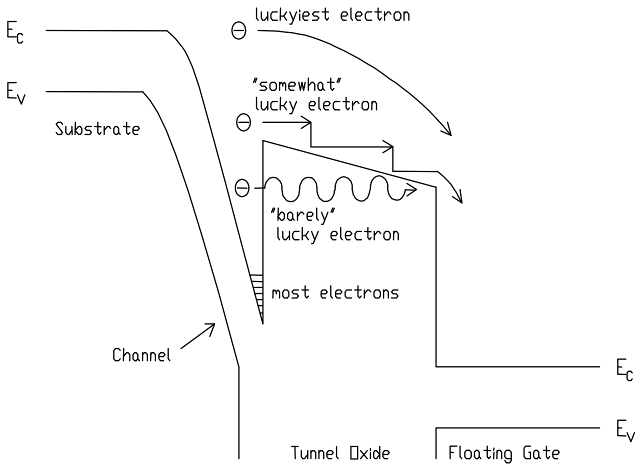

**Figure 3: Energy band diagram of programming**

**Cell Operation: Erase**

The distinguishing feature between EPROM and flash memory is the erase operation. EPROM removes electrons from the floating gate by exposure to ultra-violet light. A photon of this light source has high enough energy that if transferred to an electron on the floating gate, that electron will have enough energy to surmount the Si-$SiO_2$ energy barrier and be removed from the floating gate. This is a rather cumbersome operation requiring a UV-transmissive package and a light source. It is also rather slow and costly, often requiring the removal of the memory from the system. In flash, the contents of the memory, or charge, are removed by means of applying electrical voltages, hence to be erased in a *flash*, with the memory remaining in the system. The electrical erase of flash is achieved by the quantum-mechanical effect of Fowler-Nordheim Tunneling[3], for which the bias conditions are shown in Figure 4. Under these conditions, a high field (8-10MV/cm) is present between the floating gate and the source. The source junction experiences a gated-diode condition during erase, hence the graded source junction of the cell structure. As evidenced by the energy band diagram of Figure 5, electrons tunneling through the first ~30**Å** of the $SiO_2$ are then swept into the source. After erase has been completed, electrons have been removed from the floating gate, reducing the cell threshold. While programming is selective to each individual cell, erase is not, with many cells (typically 64k-Bytes) being erased simultaneously.

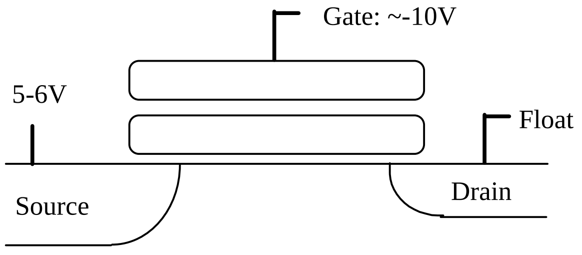

**Figure 4: Cell bias conditions during erase**

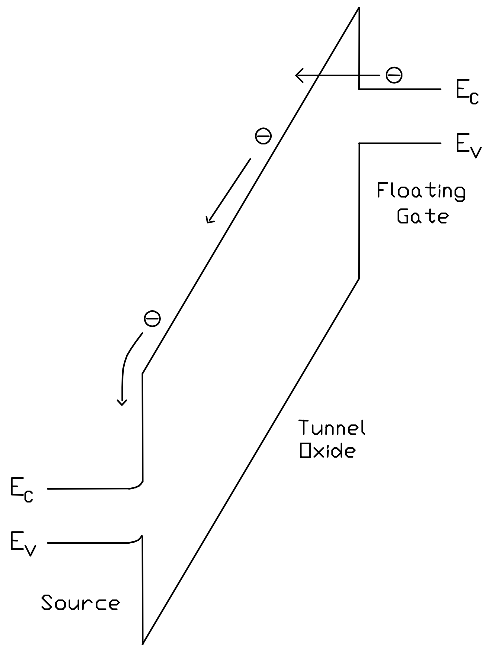

**Figure 5: Cell energy band diagram during erase**

**Cell Operation: Read**

The read operation of the cell should now be apparent. Storing electrons (programming) on the floating gate ($Q_{FG} < 0$), increases the cell Vt. By applying a control gate voltage and monitoring the drain current, the difference between a cell with charge and a cell without charge on their floating gates can be determined (Figure 6). A sense amplifier compares the cell drain current with that of a reference cell (typically a flash cell which is programmed to the reference level during manufacturing test). An erased cell has more cell current than the reference cell and therefore is a logical "1," while a programmed cell draws less current than the reference cell and is a logical "0." The floating-gate charge difference between these two states is roughly 30,000 electrons.

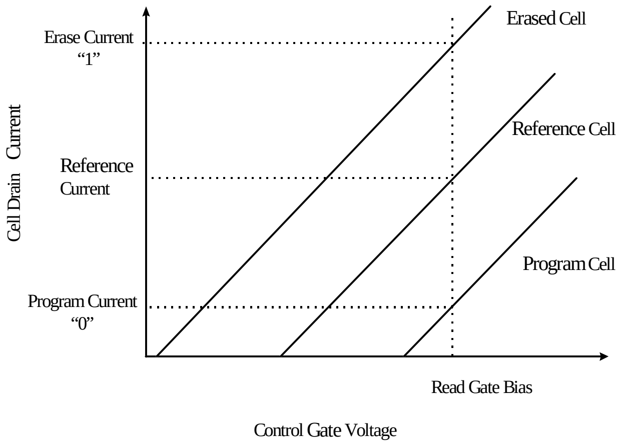

**Figure 6: Erase, program and reference cell I-V**

**Array Configuration**

Figure 7 shows a schematic drawing of the flash memory cells in a NOR array configuration. In this configuration, cells on the same wordline, or row, share common control gates. Cells on common bitlines, or columns, share common drains, which are connected via low resistance metalization, providing direct access to each cell's drain junction. The sources for cells in the array are common. They are connected locally via common degenerately doped silicon and globally via low resistance metalization. Decoders are linked to the control gate wordlines and drain bitlines to uniquely select cells at the cross point location. The direct access to the cell in this configuration versus alternative array architectures that have parasitic resistance or devices, ensures that accurate voltages can be applied to the cell and IR drops are minimized. This is a key aspect to achieving M.L.C.

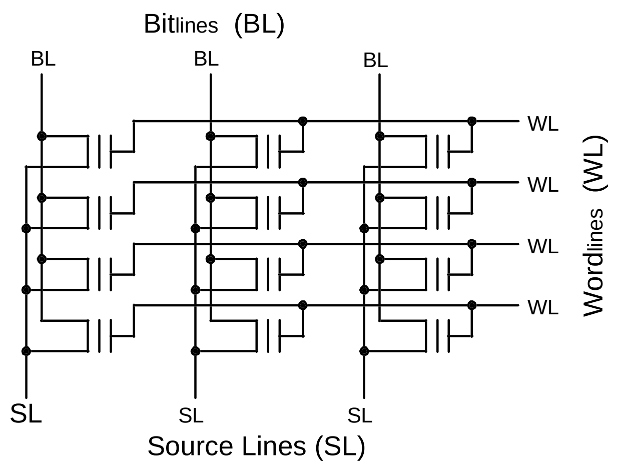

**Figure 7: Array configuration**

## M.L.C. Key Features

We have reviewed thus far how a one bit per cell (1B/C) flash memory operates. As can be inferred from the previous discussion, M.L.C. is simply a means by which charge on the floating gate is modulated and detected to levels lower than the 30,000 electrons described above, such that intermediate charge levels, or states, can be extracted from the cell. These states can now represent not just the simple 1B/C "1" and "0," but rather an M.L.C. representation with four distinct charge states: "11," "10," "01" and "00," or 2 bits in one cell. These four distinct levels are illustrated in the I-V curve of Figure 8. The key aspects of achieving these intermediate states, or levels, are precise charge placement, precise charge sensing, and precise charge retention.

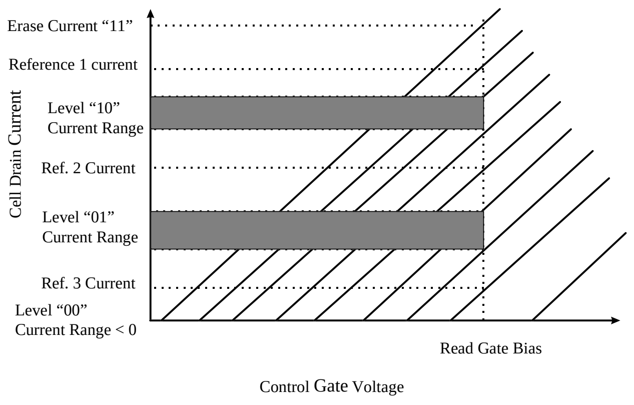

**Figure 8: Cell and reference I-V curves of 4-level 2B/C**

**Precise Charge Placement**

A comparison of Figures 6 and 8 shows that M.L.C. requires a means to control how much programming occurs within a cell. For a 1B/C product, all that is necessary is to have enough programming to change a "1" into a "0." Over-programming a cell to much higher Vt's (adding more floating gate charge) would be fine. This is not the case for M.L.C., where too much programming would cause an intermediate level to overshoot onto the next level. For instance, if a "10" was desired, but a cell was over programmed, a "01" might occur, leading to erroneous data. Therefore, a method of controlling precisely how much charge is transferred to the floating gate is required. Enough charge is needed to reach a state level without overshooting the desired level.

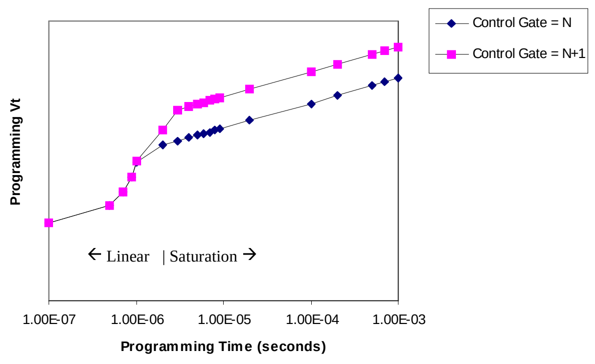

**Figure 9: Programming threshold vs. time curve**

To gain insight into how such precise control can be obtained, let's take a deeper look into the flash cell's programming characteristics. Figure 9 shows how the flash cell's Vt changes as a function of log-time under two different bias conditions. Two regions of operation are shown: linear and saturation, so-called because the linear region is linear when plotted in linear time, and the saturation region is where the cell Vt changes little with time, analogous to a MOS transistor I-V curve. Note also that in the linear region, the control gate voltage has little influence on the rate of programming, while in the saturation region, the control gate voltage has a strong dependence upon the saturated Vt. A characteristic of Figure 9 is that the flash cell programming slows as more charge is added to the floating gate. The reason for this behavior is that when in the linear region, energetic electrons, near the drain, are attracted to the floating gate. As programming progresses, the floating gate (which is coupled to the control gate and drain biases as governed by Equation 1) becomes charged more negatively, until it eventually reaches the same bias potential as the drain voltage. At this point, the energetic electrons become repelled by the floating gate charge. Programming slows, as near-drain electrons must tunnel through the $SiO_2$ barrier, or less energetic mid-channel electrons "jump" over the barrier. The strong gate dependence results from the vertical field limitation in this region. One can also see from Figure 9 that the saturated Vt increases in a one for one fashion with an increase in the programming control gate voltage. This is a simple result of the coupling Equation (1).

Given this characteristic curve, one could devise several possible methods of controlling the charge transfer to the floating gate. These methods would have to pass the criteria of being reliable (no overshoot), controllable (simple to implement), and fast (to ensure compatibility with standard flash memory product features). Programming in the linear region while being fast is not controllable. In this region, programming Vt is exponentially dependent upon time and the electron energy distribution (as determined by drain bias, channel length, doping profiles, etc.). Small variations will lead to large changes in the cell threshold and therefore overshoot of the desired state, thereby having a high likelihood of being unreliable. Minimization of these variations would be also difficult to implement. In the saturated region, the cell Vt simply depends upon the applied control gate voltage. Control in this region is more achievable. With ease of control, design optimization practices can be employed to achieve fast programming. This will be shown later. Therefore, to achieve speed and control, a placement algorithm that employed programming in the saturated region was developed.

This leaves us with reliability. Unlike Fowler-Nordheim Tunneling, used for programming in addition to erase in some versions of flash memories and subject to erratic programming due to the presence or absence of as few as one or two holes trapped in the oxide[4], channel hot-electron programming has no erratic programming mechanism. The programming threshold in saturation is simply a linear function of the applied control gate voltage. Programming in this region can be forced into an unstable operating point, known as impact-ionization induced latch-up[5]. This is the point where an excess of holes in the silicon substrate, created by the collisions of the energetic electrons with the silicon lattice, build up to the point where the parasitic NPN transistor in the silicon substrate turns on. Proper architectural design of the silicon process flow (i.e., use of EPI silicon) can easily prevent this from happening.

Therefore, the three success criteria are satisfied by exploiting stable device operation regions, namely programming in saturation. Now, a simple placement algorithm was chosen for implementation (outlined in the flowchart in Figure 10). The algorithm consists of the simple loop of programming in saturation, checking the cell Vt to determine if the desired state has been reached, stopping if the desired state is reached, or if not, incrementing the control gate voltage and providing an additional programming pulse and continuing in this fashion until the desired Vt has been achieved. In the Intel StrataFlash™ memory two bit per cell device, each programming pulse within the placement algorithm will transfer roughly 3,000 electrons of charge to the floating gate.

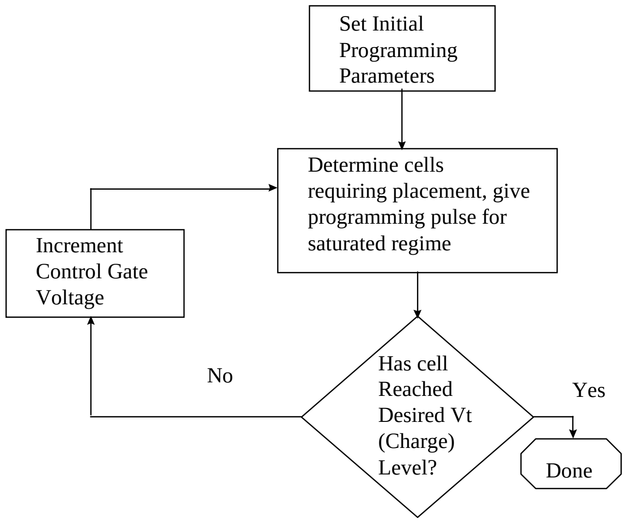
Set Initial Programming Parameters
Determine cells requiring placement, give programming pulse for saturated regime
Increment Control Gate Voltage
Has cell Reached Desired Vt (Charge) Level?
No
Yes
Done

**Figure 10: Placement algorithm flowchart**

**Precise Charge Sensing**

As can be seen from the flowchart in Figure 10, integral to the placement algorithm is a means of detecting whether or not the desired cell Vt has been achieved. Without a precise means of sensing the floating gate charge, precise charge placement would not be possible. A look back at Equation 2, the cell drain current-voltage relationship, gives some insight into what is required to achieve precise charge sensing. Control gate and drain voltage control and process Leff, Zeff, mobility and oxide capacitance control are important aspects of precise charge sensing. Drain voltage control is facilitated by direct access to the cell drain junction (bypassing any resistive IR drops) allowable in the ETOX NOR flash memory array architecture; and by applying a high enough drain voltage to operate in the saturated mode (normal MOS device saturated I-V, not programming saturation as previously discussed) where drain bias variations have minimal current impact. Process control is important, since the 33,554,432 memory cells contained within a single Intel StrataFlash memory 64Mbit device represent a >10 sigma variation, and is achieved by proper process architecture and manufacturing process control, derived from the ten years of manufacturing experience with flash memories. Control gate voltage control is achieved by an on-chip read regulation circuit, which is fully explained in a later section.

Flash memory has a unique feature associated with its non-volatility: the data write (placement) can occur under one condition of ambient temperature and system power supply, while the read out of data (sensing) can occur at a later date, at different ambient temperature and system power supply. Being fundamentally a MOS transistor, the flash cell's drain current is a function of these ambient conditions. As such, the precise charge sensing is required to span wide ranges of operation. To facilitate this needed precision, the reference levels that separate charge state levels are generated by reference flash cells contained on-chip. These reference cells, whose Vt levels are precisely placed at manufacturing test under a controlled environment, will have the same tracking with temperature and power supply as the array flash cells. This contrasts to reference levels generated by other transistor types (i.e., NMOS or PMOS), which have different temperature, voltage, and process tracking than the flash memory cell. This lessens the necessary constraints on the read regulation circuitry.

**Precise Charge Retention**

Due to the non-volatility requirement of flash memory, it is important that any charge placed on the floating gate remain intact for extended periods of time, typically ten years. This translates to a requirement of not losing more than one electron per day from the floating gate. If electron loss occurs from even one memory cell in an array of millions, the data will be corrupted. The inherent storage capability exists due to the Si-$SiO_2$ energy barrier which traps electrons on the floating gate. The inter-poly-silicon oxide (ONO film mentioned in the cell structure) is processed to maximize charge storage capabilities[6]. Under normal circumstances, the energy barrier allows charge storage for hundreds of years. There are conditions of trapped oxide charge, known as intrinsic charge loss[7], which can cause one-time shifts in threshold. These shifts are rather small and are compensated for during manufacturing test. Random defects in the insulating oxides that can lead to charge loss are less of an issue with low-defect, high-yielding process technologies, but if still present, are screened out by the manufacturing tests. These defects are driven to low enough levels on ETOX flash memories where error-correcting-codes (ECC) are not needed. The remaining concern for charge retention is any degradation to the insulating oxides that occurs as a result of the stresses of device operation.

During normal operation, high fields are applied to the flash cell. The presence of the high fields over time can degrade the charge storage capabilities of the device, by effectively lowering the energy barrier, or by providing traps sites in the oxide that can act as intermediate tunneling locations. The benefit of channel hot-electron programming, compared to tunneling for programming, is that fast programming can occur at lower internal fields thereby lessening the probability of oxide damage. Nevertheless, occurrence of damage needed to be understood to ensure the stability of the M.L.C. charge.

Consequently, the charge retention ability of the insulating oxides under various process and bias field conditions were studied in great detail. Over the course of the four year M.L.C. development period, in excess of 200 billion ($2e10^{11}$) flash cells were studied for charge retention, each to a resolution of floating gate charge of ~100 electrons. This exhaustive study provided more physical insight into the oxide damage mechanisms and has enabled us to build large scale empirical models for charge retention. The net result of this study was the ability to optimize process recipes and operating bias fields to maximize charge retention. This allows Intel StrataFlash memory to maintain high reliability performance, without the use of any ECC.

## Mixed Signal Design Implementation

The implementation of the described charge-placement algorithm and charge-sensing operation required a mixed signal circuit design of both digital and precision analog voltage generation, regulation and control circuits. The placement algorithm is executed by utilizing an on-board control engine, or the Flash Algorithmic Control Engine (FACE). FACE runs the placement algorithm by sequencing through the programming and sensing loops. During a read operation (sensing of data at a later time), the user has random access to the memory array. A read operation performs a precision-sensing operation and invokes circuitry controlling the precise cell bias voltages.

### Placement Algorithm Implementation

The placement algorithm executed by FACE is stored in a small on-chip programmable flash array. The programmable microcode allows for flexibility in algorithm changes. FACE, illustrated in Figure 11, consists of the microcode storage array, program counter (PC), arithmetic logic unit (ALU), instruction decoder, clock generator, register files, and input/output circuitry. FACE uses 6,000 transistors for logic and 32k bits of flash memory for algorithm storage.

To describe the implementation of the placement algorithm, let us assume that a group of cells (i.e., a double-word, or 32-logical bits, 16-physical cells) is to be placed and is initially in the erased state (lowest floating gate charge state). Any cells not to remain in the erased state (representing logical data "11") will receive a programming pulse. FACE will look up the drain and initial control gate voltage stored in a permanent read-only register located on-chip. FACE will then set the control gate voltage through the digital to analog converter (DAC). The DAC circuit receives the FACE digital input and divides the on-chip generated 12-volt power supply (VP12) to achieve the desired control gate voltage for that particular programming pulse. The drain voltage, used during the programming pulse, is generated from a regulation circuit that sets the gate voltage on a source follower. FACE will continue to supply the programming voltages for the pre-determined amount of time sufficient to reach the saturation region. When the programming pulse is complete, FACE will reconfigure the circuits to perform the sensing portion of the algorithm, an operation called verification. The drain and control gate voltages are now set to the same values as used in a user read access to ensure common mode between verification and read. FACE will take the result of the verification and determine which cells have reached their destination charge level and which have not. Those that have not will require an additional programming pulse with an increased control-gate voltage. A cell that no longer requires additional programming pulses will have the drain voltage disabled by the program pulse selector circuit. This sequence of events continues until all cells in the double-word have completed programming.

### Analog Circuit Blocks for Precise Charge Placement

Placement requires precision voltages covering a range of 4-12 volts, while the chip Vcc (user supplied voltage) is kept at a typical value of 5 volts. The voltages applied to the memory array need to be internally generated and precisely regulated. On-chip voltage generation is achieved by use of charge pumps, in which switched capacitors boost the user-supplied Vcc to higher values. Voltages are controlled using a precision voltage reference circuit and voltage regulation circuits (Figure 11).

During a programming pulse, two charge pumps are used. One charge pump generates the internal 12V supply (VP12). This is used to supply a precision control gate voltage to the flash cells, through the DAC circuit.

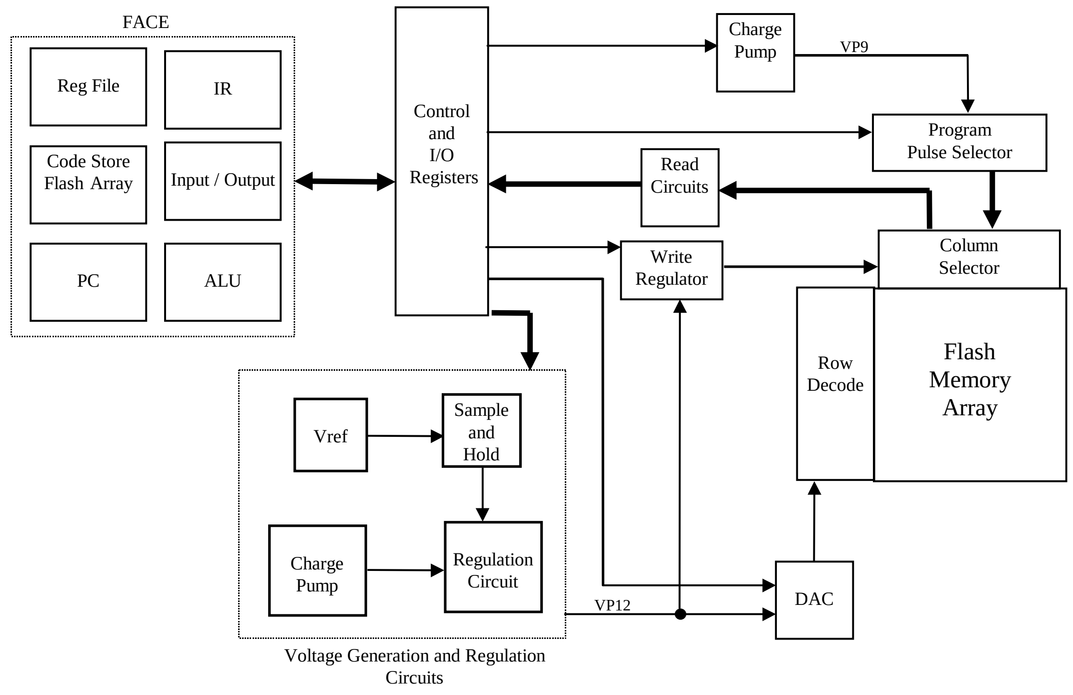

**Figure 11: FACE and placement operation block diagram**

VP12 also serves to generate the precision flash drain voltage through the write regulation circuit (WRC). The WRC generates a voltage that is applied to an NMOS transistor configured as a source follower. This transistor is in the bitline (or drain) path of the flash cell. The flash cell drain current is supplied through a second pump that generates the signal VP9. This pump is required to supply the programming current for up to 32 flash cells at a time.

During the placement algorithm, voltage stability is critical to precise charge storage. Any variations in the reference circuit voltages will be seen as variations in the flash control gate voltage, to which the programming saturated Vt is directly related. To achieve this absolute stability in the voltage reference circuit, a sample and hold circuit is employed. At the start of the placement algorithm, the sample and hold circuit samples the reference voltage and holds the value on a capacitor during the running of the entire algorithm. This guarantees the control gate voltage varies from pulse to pulse by only the desired step value and not by any additional components.

**Circuit Blocks for Precise Charge Sensing**

When the device is in the read mode of operation, FACE is disabled and the user has control to access the memory array. A read operation consists of sensing 16 bits worth of data from a random location in the memory array. With M.L.C., 8 flash cells are used to obtain 16 bits of data. During the read operation (Figure 12), the flash cell control gate voltage is controlled through a read regulator circuit (RRC). Minimizing this voltage variation will minimize the variations in cell current (Equation 2). This allows for more precise measurement of the charge level stored on the floating gate. Drain voltage stability is also important to ensure that the flash cell being sensed has a high enough drain voltage to keep the memory transistor operating in the saturated region of the MOS I-V.

Due to fluctuations in user supplied Vcc and a lower value than may be needed during read, an internal voltage charge pump is used during a read operation to generate the internal voltage to supply the flash cell control gate. The RRC uses the same voltage reference circuit that is used for voltage regulation during a placement operation, as mentioned above. However, in the case of a read operation, not as much voltage stability is required so the sample and hold circuitry is not used.

**Parallel Charge Sensing**

High speed random access and precise charge sensing are accomplished through a parallel charge-sensing scheme. Through direct connections to each memory cell, the data read operation determines the level of each memory cell quickly, accurately, and reliably. The data read operation senses which of the four levels the memory cell falls within based on the threshold voltages of three reference cells. This is done simultaneously with three sense amplifiers (Figure 13), where each sense amplifier compares the flash cell current being sensed to

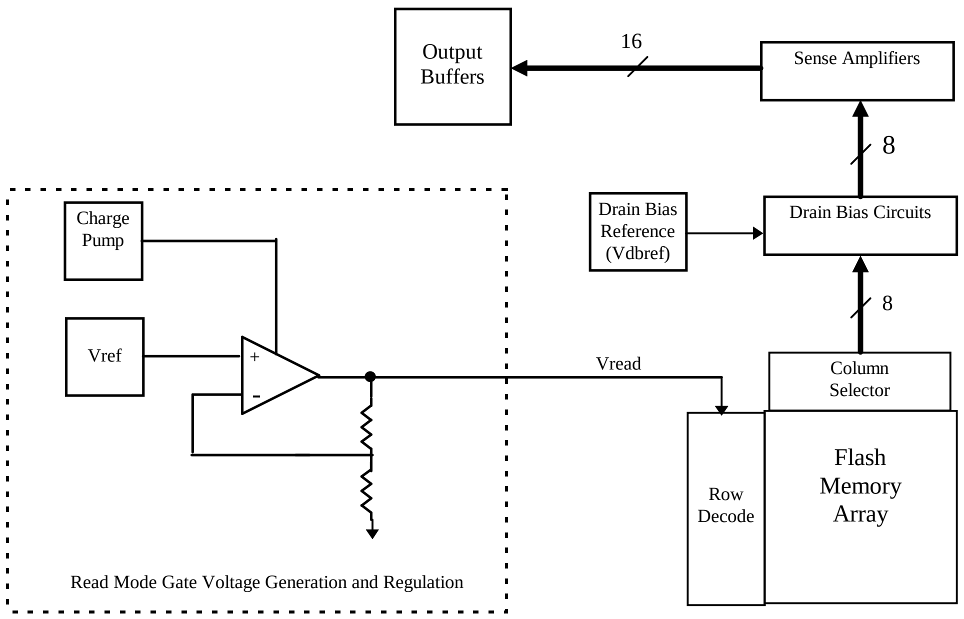

**Figure 12: Read operation block diagram**

the current of the flash reference cells.

The memory cell and the reference cells are biased in such a way that each conducts a current (Icell and Iref) proportional to their respective threshold voltage (Vt and VtRef). During a read operation, Vread is placed on the control gates of the memory and reference cells, the source terminals are grounded, and the drain voltages are set through a bias circuit that utilizes a precision voltage reference circuit.

The current for the memory cell being sensed is compared to the current of the three reference cells. The memory cell and reference cell current is converted to a voltage through an active load transistor. The resultant voltages are compared by the three sense amplifiers. A sense amplifier is associated with each of the three reference cells. Each sense amplifier also has an input from the flash cell being sensed. If the current of the cell being sensed is greater than the current of the reference cell (Icell > Iref or Vt < Vtref,), the sense amplifier output is a logic "1." If the current of the cell being sensed is less than the current of the reference cell, the sense amplifier output is a logic "0." The outputs of the three sense amplifiers are connected to a logic circuit that interprets the two data bits in parallel.

| Cell Vt | Output Sense Amp 1 | Output Sense Amp 2 | Output Sense Amp 3 | D1 | D0 |
|---|---|---|---|---|---|
| Vt < VtR1 | 1 | 1 | 1 | 1 | 1 |
| VtR1 < Vt <VtR2 | 0 | 1 | 1 | 1 | 0 |
| VtR2 < Vt <VtR3 | 0 | 0 | 1 | 0 | 1 |
| Vt >VtR3 | 0 | 0 | 0 | 0 | 0 |

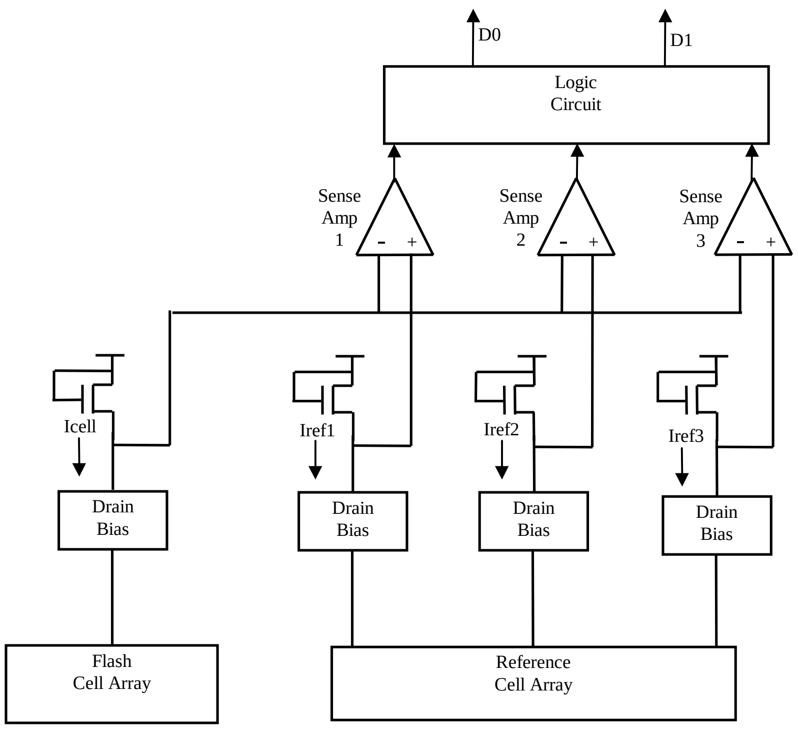

**Figure 13: Parallel charge sensing**

## Low-Cost Design Implementation

Traditionally, a storage element in a memory corresponds to one bit of information. To double the amount of memory, the memory array or memory storage elements would need to be doubled. In addition to doubling the number of memory elements in the array, certain memory interface circuits must also be doubled. In particular, the memory array needs to be decoded requiring wordline and bitline decoders. In a typical single transistor non-volatile memory device (flash, EPROM), approximately 20% of the silicon area used is due to these interface circuits required to access the array. These interface circuits typically do not scale with process technology at the same rate as the memory array because they have high voltage and analog requirements.

Intel StrataFlash memory doubles the storage capacity of a memory device without doubling the memory array and the associated interface decoding circuitry. Additional circuitry is required to achieve the multiple bits per cell, but takes up a relatively small additional area. The additional overhead for circuitry is due mostly to the additional sense amplifiers, reference circuitry, and circuitry for voltage generation or charge pumps. The additional silicon area required for this circuitry represents only an additional 5% over what is necessary for a one bit per cell device. Implementations of M.L.C., which require externally supplied components (i.e., microcontroller, ECC, and voltage regulators), have the cost savings of M.L.C. diminished by these peripheral overheads. Intel StrataFlash memories achieve 2x the density at very close to 1x the area.

## Low-Cost Process Manufacturing

ETOX flash memory has a long manufacturing history. As such, it was necessary that any implementation of M.L.C. not disrupt that history by having unique process requirements, which would cause a slow yield learning period or poor manufacturing throughput. First and foremost for M.L.C. to be successful, it must be able to ride on a technology that produces error-free one bit per cell flash memory. This requirement throughout Intel's ETOX NOR flash memory's history has resulted in tight manufacturing margins and the learning necessary for achieving such margins. Memories that rely on ECC for even one bit per cell operation have little margin built into the basic technology. Throughout the previous discussions, mention has been made of process manufacturing attributes for M.L.C. These attributes have been achieved by utilizing the same process flow as the standard one bit per cell flash memory. This approach has maintained shared learning and has lead to lower costs. In other words, low-cost process manufacturing was achieved through an understanding of M.L.C. requirements up-front in the design of the basic process architecture at the generation where M.L.C. is introduced. The tight manufacturing margins required for M.L.C. are a natural extension of the learning from manufacturing of error-free one bit per cell flash memory and are well within the manufacturing, equipment, and process module capability.

## Standard Product Feature Set

One of the main challenges in implementing M.L.C. is maintaining product performance, usability, and reliability at the same levels as standard flash memories. If the implementation of M.L.C. resulted in a product that did not satisfy these goals, it would be relegated to a niche in the marketplace. Key features for a non-volatile memory are programming speed, read speed, power supply requirements, and reliability. This paper shows how our implementation of M.L.C. achieves these features. Before finishing, however, let us briefly discuss each one of them.

### Programming Speed

Programming speed is achieved by choosing a placement algorithm that exploits stable device operating points to enable circuit performance optimization to occur, with little limitations of flash device operation. Parallel cell programming (32 cells, or 64 bits) at a time also amortizes the placement algorithm run time. The choice of charge sensing approaches also affects programming speed as it is integral to the placement algorithm. Sensing approaches other than those described in this paper can be used. An example would be a sensing scheme that varies control gate voltage to detect the threshold voltage directly. Such a scheme, while a more direct measure of floating gate charge, does not exploit the current drive capability of the flash cell, the drive used for sensing speed performance. To sum up, the choices of algorithms, optimizations, and architecture are what allow M.L.C. programming to be as good or better than one bit per cell flash memories.

### Read Speed

As mentioned above, the choice of fixed control gate sensing and utilization of the flash cell's current drive capability allows fast read operation. In addition, parallel charge sensing allows for fast decode of the logic level, with little circuit overhead. As such, the read speed of Intel's StrataFlash memory is consistent with that of one bit per cell flash memories of comparable bit density.

**Power Supply**

As also discussed, the on-chip voltage generation and regulation is key to the implementation of M.L.C. One could specify an M.L.C. product that uses externally supplied precision voltages, but such a product would be more costly to the user, who would have to pay for the power supply, memory, and board space. Having the voltages generated and regulated on-chip allows for the Intel StrataFlash memory to plug directly into existing flash memory applications.

**Reliability**

Starting with high-yielding, low-defect memory, exhaustive cell studies and process and bias optimizations allow for an implementation of M.L.C. that achieves non-volatility and high reliability without requiring on-chip or system ECC. Thus the user can interface to the device with random memory location access, without latency for correction. Additionally, ECC requires overhead bits, which would diminish the cost advantages of M.L.C.

These standard flash memory product features, coupled with low-cost circuit design and manufacturing process implementation allow users to benefit from the low cost of M.L.C. without having to sacrifice needed features or performance.

## Conclusion

It has been shown how Intel StrataFlash memory achieves multiple bits per cell, coupled with traditional process scaling, to provide an advance in memory cost reduction. The M.L.C requirements of precise charge placement, precise charge sensing and precise charge retention are achieved by exploiting stable device-operating points and direct access to the memory cell, employing mixed signal digital and analog design. Non-cell-related costs are held low by riding on the tight manufacturing margins developed for error-free one bit per cell flash memories. A standard product feature set ensures that the cost advantages of M.L.C. are available to the mainstream flash memory market.

## Acknowledgments

The authors would like to thank the members of the M.L.C. development groups, whose dedicated work helped turn a few ideas into a product reality.

## References

[1] Kynett, V.N., et. al., "An In-System Reprogrammable 256K CMOS Flash Memory," Technical Digest IEEE International Solid State Circuits Conference, 1988, pp. 132-133.

[2] Tam, S., Ko, P.K., and Hu, C., "Lucky-Electron Model of Channel Hot Electron Injection in MOSFET's," IEEE Transactions Electron Devices, September 1984.

[3] Lenzlinger, M. and Snow, E.H., "Fowler-Nordheim Tunneling into Thermally Grown $SiO_2$," Journal of Applied Physics, vol. 40, No. 1, January 1967, pp. 278-283.

[4] Ong, T.C., et. al., "Erratic Erase in ETOX$^{TM}$ Flash Memory Array," IEEE VLSI Symposium, 1993, p. 145.

[5] Eitan, B. and Frohman-Bentchkowsky, D., "Surface Conduction in Short-Channel MOS Devices as a Limitation to VLSI Scaling," IEEE Transactions on Electron Devices, vol. ED-29, No. 2, February 1982, pp. 254-266.

[6] Wu, K., et. al., "A Model for EPROM Intrinsic Charge Loss Through Oxide-Nitride-Oxide (ONO) Interpoly Dielectric," 28th Annual Proceedings IEEE International Reliability Physics Symposium, 1990, p. 145.

[7] Mielke, N., "New EPROM Data-Loss Mechanisms," 21st Annual Proceedings IEEE International Reliability Physics Symposium, 1983, p. 106.

## Authors' Biographies

Al Fazio is a Principal Engineer in Flash Technology Development. He received a B.Sc. in Physics from the State University of New York at Stony Brook in 1982 and joined Intel the same year. He has been involved in development programs including SRAM, EPROM, $E^2$PROM, NVRAM, and Flash Memories. He was responsible for the Technology Development of the Intel StrataFlash$^{TM}$ memory. He holds more than a dozen patents and has authored or co-authored several technical papers, two of which have received Outstanding Paper Awards at the IEEE International Reliability Physics Symposium and at the IEEE International Solid State Circuits Conference. He is presently responsible for Intel's Multi-Level-Cell and Advanced Flash Memory Cell Development and currently serves as General Chairman of the IEEE Non-Volatile Semiconductor Memory Workshop. His e-mail address is al_fazio@ccm.sc.intel.com

Mark Bauer is a Senior Staff Engineer in Flash Circuit Design. He received his B.S.E.E. from the University of California, Davis in 1985. He joined Intel's Memory Components Division that same year, working on EPROM design. He was responsible for Circuit Design Development of the Intel StrataFlash$^{TM}$ memory. He holds more than a dozen patents in the field of non-volatile memories and has authored two technical papers, one of which received an Outstanding Paper Award at the IEEE International Solid State Circuits Conference. He is presently responsible for Intel's next generation Multi-Level-Cell Circuit Design. His e-mail address is mark_bauer@ccm.fm.intel.com

# Redundancy Yield Model for SRAMS

Nermine H. Ramadan, STTD Integration/Yield, Hillsboro, OR, Intel Corp.

Index words: Poisson’s formula, yield, defect density, repair rate

## Abstract

This paper describes a model developed to calculate the number of redundant good die per wafer. A block redundancy scheme is used here, where the entire defective memory subarray is replaced by a redundant element. A formula is derived to calculate the amount of improvement expected after redundancy. This improvement is given in terms of the ratio of the overall good die per wafer to the original good die per wafer after considering some key factors. These factors are memory area, available redundant elements, defect density and defect types with respect to the total reject die and defect distribution on the memory area. The model uses Poisson’s equation to define the yield, then the appropriate boundary conditions that account for those factors are applied. In the case of a new product, knowing the die size, memory design, and total die per wafer, the model can be used to predict the redundancy yield for this product at different initial yield values. Optimizing the memory design by varying the number of memory blocks and/or redundant elements to enhance redundancy is also discussed. The model was applied to three products from two different process generations and showed good agreement with the measured data.

## Introduction

Due to the continuing increase in the size of memory arrays, reaching a high yield from the same wafer is more challenging than ever. Redundancy is a way to improve the wafer yield and to reduce the test cost per good die by fixing potentially repairable defects. In order to forecast the volume of a certain product when redundancy is applied, it is important to estimate, as accurately as possible, the number of die gained after redundancy.

Redundancy is the process of replacing defective circuitry with spare elements. In SRAMs, rows and/or columns can be replaced, as well as an entire subarray. In a previous study[1], a redundant yield estimation methodology was developed. It is applicable to row, column or block redundancy schemes. It distinguishes between repairable and non-repairable faults within a memory block. In order to apply this method, new CAD tools are required. This method is useful if row or column redundancy is used.

This paper will focus only on the yield estimation for block redundancy, as block redundancy was preferred over row and column redundancy for the SRAM architecture. It is usually easier to replace the entire subarray. This might seem like overkill; however, replacing the entire subarray allows for the replacement of defective peripheral circuits in addition to just the memory array elements. It also allows for the replacement of multiple bad bits, or other combinations of failing bits, rows and columns.

A yield multiplier M is defined as the ratio of the total good die after redundancy to the original good die per wafer, or

$$M = \text{total redundant good/original good} \qquad (1)$$

so that the redundant yield , $Y_{red}$ , is given as

$$Y_{red} = M \times Y \qquad (2)$$

where Y is the initial yield. Forecasting of the redundancy yields is based on how accurately the factor M is calculated. A formula for M was obtained by using the correlated defect model. According to this model, an expression for the yield of die containing a number of defects, I, is given by

$$y_I=\{(n+I+1)! \times (DA)^I\}/\{(I!\, n^{I-1})\times(1+D\,A/n)^{n+I}\} \times f^I \qquad (3)$$

where

$y_I$ = yield of a die with I defects

D = average defect density ( #/$cm^2$ )
A = die area ($cm^2$ )
n= correlation factor between defects
f = fraction of the die area that contains the defects

The yield of die with zero defects can be obtained by setting I = 0 and f = 1 as

$$Y = 1 / \{ 1 + (A D / n) \}^n \qquad (4)$$

With n = 4 and using equation (4) to substitute for the defect density, equation (3) becomes

$$y_I = Y \times ((I+3)(I+2)(I+1) /6) \times f^I \times ( 1-Y^{1/4})^I \qquad (5)$$

Introducing g as the fraction of repairable defects, g varies depending on the number of repaired defects. An expression for M was obtained by summing $y_I$ over the ratio of correctable defects and substituting in (2)

$$M = 1 + {}^k\Sigma_{I=1} ((I+3)(I+2)(I+1) /6) x( g f ( 1-Y^{1/4}))^I \qquad (6)$$

M was calculated by entering arbitrary values of g and f in equation (6). However, there was no evidence to support the values of the repairable defect density represented by g used to calculate M.

Another formula was used to estimate the yield multiplier M . The yield is derived from Poisson's equation [2]

$$Y = \exp ( -AD ) \qquad (7)$$

Instead of using a constant defect density, D, Murphy assumed several defect density distributions[3]. The most preferred distribution was a Gaussian. Stapper used a gamma distribution, which led to the following yield formula [4]

$$Y = 1 / \{ 1 + (A D / \alpha) \}^\alpha \qquad (8)$$

where α is the average value of the coefficient of variation for the gamma function. The yield multiplier derived from the previous yield formula is given by

$$M = S \times ( 1 + 0.01 ( L + I ) A_{sb} D / k )^k \qquad (9)$$

where
S = fuse programming success rate
I = number of redundant elements
L = number of subarrays
$A_{sb}$ = area of subarray ( $mm^2$ )
k = constant for MOS process
0.01=conversion from $mm^2$ to $cm^2$

This simple formula is actually overestimating the redundancy improvement, since it assumes that all the defects are repairable.

In order to get a better estimate of the yield improvement, the nature and distribution of the defect need to be understood. These are taken into account in this model. When considering defects, it is important to realize that not all reject die are repairable: a die failing for a short, for example, cannot be repaired. Also the number of defective subarrays that could be repaired depends on the available redundant elements per memory block. This means that having more than one defect per die requires a certain distribution of those defects in order for redundancy to be successful.

Taking into account the above factors and using Poisson's equation to describe the yield, the present model was able to predict the redundant yield within the same range as shown by the real data. The following section illustrates how the key parameters affecting redundancy are used to develop the model.

## SRAM Array Layout

Figure 1 shows the layout of a SRAM memory array. Before going into details, the following terms are defined as they will be used throughout the paper:

- *Subarray*. This is a unit array of the memory area, and is shown as subarrays 0 to 72 in Figure 1.
- *Memory block or block*. This is a segment of the memory area, and is one of four rows shown in Figure 1.
- *Redundant element or element*. This is a spare subarray used to replace a memory subarray, and is given as subarrays R in Figure 1.

The die consists of two areas:

- *Repairable area*. This includes all the circuitry in the subarray. In this model the repairable area is the sum of the areas of the memory subarrays.
- *Non-repairable area*. This includes the periphery area. The redundancy elements are also considered part of the non-repairable area.

Block redundancy is illustrated in Figure 1. The defective subarray "4" is replaced by the redundant element R in the same memory block. This is done by programming the right fuses and shifting the array assignments.

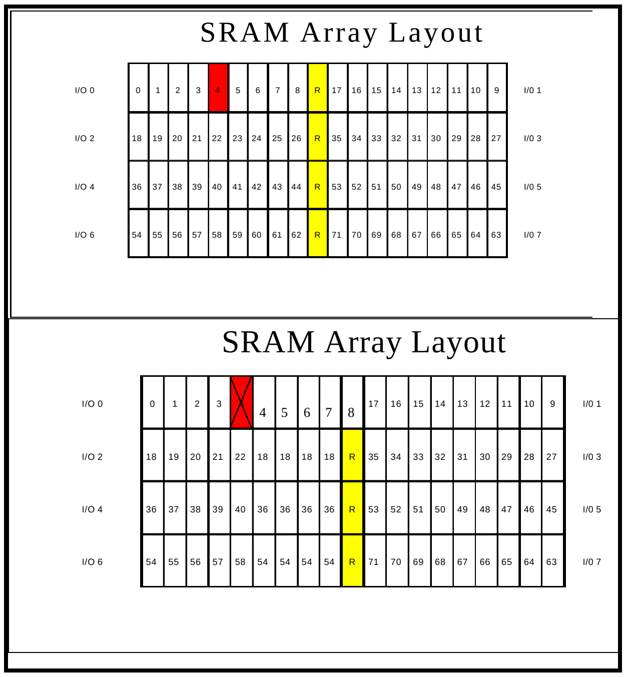
SRAM Array Layout
I/O 0
I/O 1
I/O 2
I/O 3
I/O 4
I/O 5
I/O 6
I/O 7
0 1 2 3 4 5 6 7 8 R 17 16 15 14 13 12 11 10 9
18 19 20 21 22 23 24 25 26 R 35 34 33 32 31 30 29 28 27
36 37 38 39 40 41 42 43 44 R 53 52 51 50 49 48 47 46 45
54 55 56 57 58 59 60 61 62 R 71 70 69 68 67 66 65 64 63
SRAM Array Layout
0 1 2 3 4 5 6 7 8 17 16 15 14 13 12 11 10 9
18 19 20 21 22 18 18 18 18 R 35 34 33 32 31 30 29 28 27
36 37 38 39 40 36 36 36 36 R 53 52 51 50 49 48 47 46 45
54 55 56 57 58 54 54 54 54 R 71 70 69 68 67 66 65 64 63

**Figure 1: SRAM array layout used in the model**

## Definitions

- Yield Equation:

The yield model used here is Poisson's yield model [2]

$$P_n = \{ \lambda^n \exp(-\lambda) \} / n! \qquad (10)$$

where

$P_n$ = the probability of n defects on a die of area A and defect density D

$\lambda$ = A D

Defining the failure probability as the probability that a die has one or more defects as

$$F_n = \sum_1^\infty \{ \lambda^n \exp(-\lambda) \} / n! = 1 - \exp(-\lambda) \qquad (11)$$

and defining the yield as the survival probability

$$S_n = 1 - F_n \qquad (12)$$

the yield equation is then

$$Y = \exp(-\lambda) = \exp(-AD) \qquad (13)$$

- Improvement Factor:

The improvement factor is defined as

$$M = (Ngd + Nrep) / Ngd \qquad (14)$$

where

Ngd = initial number of good die per wafer

Nrep = number of repaired die per wafer

Using Poisson's formula to derive an expression for Ngd and Nrep, Ngd can be defined as the number of die with zero defects

$$Ngd = N \exp(-\lambda) \qquad (15)$$

where N is the total number of tested die. An expression for Nbd is given by

$$Nbd = N(1 - \exp(-\lambda)) \qquad (16)$$

which is the number of die with one or more defects. Assuming all bad die are repaired, Nrep is then equal to Nbad, and a formula for a maximum improvement factor is given by

$$M_{max} = \exp(-\lambda) \qquad (17)$$

However, this in fact is not the case; therefore, Nrep needs to be represented by a more realistic formula. The model described in this paper started with a simple assumption and more details were gradually added to get as close as possible to the real case. The following section illustrates this development.

## Yield Improvement Formula

### Simple Model

As a first approach, Nrep is represented by the number of die with one defect, and Poisson's formula is used again to describe Nrep(1) as

$$Nrep(1) = N \lambda \exp(-\lambda) \qquad (18)$$

Substituting the improvement factor formula, equation (14)

$$M_{simp} = 1 + \lambda \qquad (19)$$

As mentioned before, not all the die area could be fixed; only the memory area was repairable. Instead of A, the total area, $A_{rep}$, is used in the expression of $\lambda$, where

$$A_{rep} = F_{area} \times A \qquad (20)$$

and $F_{area}$ is the fraction of the repairable area. Since only random defects can be fixed by redundancy, the random defect density is used in the expression of λ, where D is found from the yield equation, equation (13), as

$$D = -\ln Y / A \qquad (21)$$

Here Y is the random yield and is given by Y = Ngd/N and is calculated from the data. Using $A_{rep}$ instead of A, the expression for λ that will be used for the rest of the analysis is then

$$\lambda = F_{area} \times A D \qquad (22)$$

**Cumulative Model**

Next, a better definition of Nrep is obtained: the number of die with one, two, or n defects. Poisson's equation is used to derive a formula for the number of die with a certain number of defects. Since

$$P_n = \{ \lambda^n \exp(-\lambda) \} / n! \qquad (10)$$

where n is the number of defects, the following improvement factors can be defined:

$M_1$ = improvement factor from die with one defect
$M_2$ = improvement factor from die with two defects
$M_n$ = improvement factor from die with n defects and is equal to

$$M_n = 1 + \Sigma \, Nrep_n / Ngd \qquad (23)$$

where

$$\Sigma \, Nrep_n = N \, \Sigma \{ \lambda^I \exp(-\lambda \} / I! \qquad (24)$$

from I = 1 to I = n

The improvement factors are then given by

$M_1 = 1 + \lambda$……………………………….1 def/die
$M_2 = 1 + \lambda + (\lambda)^2/2!$……………………...2 def/die

and for n defects per die

$$M_n = 1 + \lambda + (\lambda)^2 / 2! + (\lambda)^3 / 3! + \ldots + (\lambda)^n / n! \qquad (25)$$

Since there is a possibility of having more than one defect per block, λ must be multiplied by a so-called repair probability $R_n$, where $R_n$ is the ratio of the combination of blocks and defects that can be repaired to the total number of combinations. This depends on the available number of redundant elements. $M_n$ is then written as

$$M_n = 1 + R_1 \lambda + R_2 (\lambda)^2 / 2! + R_3 (\lambda)^3 / 3! + \ldots + R_n (\lambda)^n / n! \qquad (26)$$

An expression for $R_n$ is found by using a binomial series expansion. If G = X + Y, where X and Y represent the number of blocks, the resulting binomial series is

$$G^n = (X + Y)^n = X^n + n X^{n-1} Y + {}^nC_2 X^{n-2} Y^2 + \ldots + {}^nC_k X^{n-k} Y^k + Y^n \qquad (27)$$

with

$${}^nC_k = n! / k! (n-k)! \qquad (28)$$

as the coefficient of X. Note that this coefficient represents the number of terms with X raised to a certain power, where this power represents the number of defects on this block.

If G contains more than two terms, or more than two blocks, G is the written as

$$G = X + Y + Z + \ldots \text{ up to b blocks}$$

and the series becomes

$$G^n = (X + Y + Z + ..)^n = X^n + n X^{n-1} Y + n X^{n-1} Z + n X^{n-1} \ldots + {}^nC_2 X^{n-2} Y^2 + {}^nC_2 X^{n-2} Z^2 + \ldots + {}^nC_k X^{n-k} Y^k + {}^nC_k X^{n-k} Z^k + \ldots + Y^n + Z^n + \ldots \qquad (29)$$

Knowing that each redundant element, e, can fix one defect, a term raised to the power of e+1 or higher indicates that it has more defects than elements and it cannot be fixed. This means that the number of possibly repaired blocks is equal to the total number of blocks and defect combinations minus the sum of coefficients of the terms raised to the power of e+1 or higher. All terms can be treated similarly, since all blocks are equal, and terms raised to the same power are collected together. Their coefficients can then be added together as well. Each coefficient in the previous series is repeated b-1 times for b terms. Except for the highest power in the series, it exists only b times. This means that the sum of coefficients can be written as

$$\text{sum} = \Sigma \, b (b-1)^{(n-k)} \, n! / k! (n-k)! \qquad (30)$$

from k=1 to k = n, the number of defects. The number of repairable blocks is then

$$G_{rep} = (b)^n - \Sigma \, b (b-1)^{(n-k)} \, n! / k! (n-k)!$$

from k = e+1 to k = n and $n \geq k$ always

From the definition of $R_n$, the total combination of blocks and defects can be given by $b^n$. The repair probability is the ratio of the possibly repaired count to the total count, or

$$R_n = G_{rep} / b^n$$
$$= \{(b)^n - \Sigma\, b\,(b-1)^{(n-k)}\, n!\,/\,k!\,(n-k)!\,\}/(b)^n \quad (31)$$

and the formula for the cumulative improvement factor is

$$M_n = 1 + R_1 \lambda + R_2 (\lambda)^2 / 2! + \ldots + R_n (\lambda)^n / n! \quad (32)$$

Note that this formula is applicable to up to e x b defects, which is the total number of elements and blocks; beyond that it is not useable. Higher order terms in the series are also negligible and can be ignored without affecting the improvement factor.

**General Model**

A general model is developed by including the effect of defect type in the previous improvement factor formula. The cumulative model is in fact overestimating the real data, because it assumes that all die are repairable. Studying the reject die data, it was found that only certain die could be fixed, namely raster type bins which occupy a certain fraction of the total reject die population. Adding to this the other restriction of obeying the previously described repair probability, only a certain fraction of those die is repairable. An efficiency factor $\eta$ is introduced into the cumulative model. It is defined as the effective fraction of bad die repaired extrapolated at the maximum yield for a certain repairable area. It is calculated from

$$\eta = \gamma / F_{area} \quad (33)$$

where

$\gamma = (Nbd\_cr / Nbd) \times (Nrep / Nbd\_cr)$
Nrep = number of repaired die
Nbd = number of reject die
Nbd_cr = correctable reject die

which cancels out in the expression of $\gamma$. $\lambda$ is then modified to

$$\lambda = \eta F_{area} \times A\ D \quad (34)$$

which is then used in the general model

$$M_n = 1 + R_1 \lambda + R_2 (\lambda)^2 / 2! + \ldots + R_n (\lambda)^n / n! \quad (35)$$

This is the same as the cumulative model formula, equation (32), except for the expression of $\lambda$. $\gamma$ is obtained from the empirical data, so that one value of $\gamma$ can be used for products from the same process generation. For a new process, $\gamma$ from a previous process can be used, since its value is close from one process to the other.

## Redundant Elements and Memory Blocks Optimization

In this section, how the number of redundant elements and memory blocks affects the yield improvement is studied. Increasing the number of spare elements increases the chance of repair. However, this impacts the repairable area, since the total area increases, while the repairable area is fixed. The dependency of the improvement factor on both the number of redundant elements and the repairable area is studied in order to check the possibility of improving redundancy by varying these two factors.

Since the improvement factor is a function of the repairable area and the defect density, and since the defect density is also a function of the area as calculated from the yield equation, equation (21), this equation is used to substitute for D in the expression for $\lambda$, equation (34), as

$$\lambda = \eta F_{area} \times (A\ D) = -\eta F_{area} \times \ln Y \quad (36)$$

The total die area is then written as

$$A = A_{rep} + A_{nrep} + 4 \times A_{el} \quad (37)$$

where $A_{rep}$ is the repairable area and is equal to the area of the subarrays, $A_{nrep}$ is the area of the die circuitry, and $A_{el}$ is the redundant element area and is equal to the subarray area. Increasing the number of redundant elements by sets of 4, the total area is

$$A = A_{rep} + A_{nrep} + 4 \times e \times A_{el} \quad (38)$$

where "e" is the number of elements/block. The fraction of the repairable area is then

$$F_{area} = A_{rep} / (A_{rep} + A_{nrep} + 4 \times \mathbf{e} \times A_{el}) \quad (39)$$

and

$$\lambda_i = -\eta A_{rep} \ln Y / (A_{rep} + A_{nrep} + 4 \times I \times A_{el}) \quad (40)$$

To study the behavior of the improvement factor with e and $A_{rep}$ , start with the general yield improvement factor formula, equation (35)

$$M_n = 1 + R_1 \lambda + R_2 (\lambda)^2 / 2! + R_3 (\lambda)^3 / 3! + \dots + R_n (\lambda)^n / n!$$

In the case of adding extra redundant elements, the die area, and hence $\lambda$, is also changing, so that for each case with a certain number of elements the value of $\lambda$ is different. The improvement factor formula is written as

1 element, n defects:
$$M_{n,1} = 1 + R_1 \lambda_1 + R_2 (\lambda_1)^2 / 2! + R_3 (\lambda_1)^3 / 3! + \dots + R_n (\lambda_1)^n / n!$$

2 elements, n defects:
$$M_{n,2} = 1 + R_1 \lambda_2 + R_2 (\lambda_2)^2 / 2! + R_3 (\lambda_2)^3 / 3! + \dots + R_n (\lambda_2)^n / n!$$

e elements, n defects:
$$M_{n,k} = 1 + R_1 \lambda_e + P_2 (\lambda_e)^2 / 2! + \dots + R_n (\lambda_e)^n / n! \qquad (41)$$

For the number of defects less than or equal to the number of elements per block, the die is always repairable, i.e., $R_n = 1$ for all terms with $n \le e$.

On the other hand, if a die has n defects, where $n > e \times b$, the die is never repairable. The improvement factor formula is then written as

1 element, $n \le 1$ defects:
$$M_{n,1} = 1 + \lambda_1$$

2 elements, $n \le 2$ defects:
$$M_{n,2} = 1 + \lambda_2 + (\lambda_2)^2 / 2!$$

e elements, $e < n \le e \times b$ defects:
$$M_{n,e} = 1 + \lambda_e + (\lambda_e)^2 / 2! + \dots + (\lambda_e)^e / e! + R_{e+1} (\lambda_e)^{e+1} / e+1! \dots + R_n (\lambda_e)^n / n! \qquad (42)$$

where $R_n$ follows the expression given by equation (31).

Next the effect of increasing the number of blocks per memory area on the redundant yield is studied. Dividing the memory area into a larger number of blocks also increases the chance for repair, since each block is accompanied by one redundant element. However, there is a certain maximum number of blocks, after which the increase in improvement is negligible, since the larger order terms in the series start to diminish. In this analysis, the total number of subarrays and $F_{area}$, are kept constant, but the size of the subarray is changed depending on how the memory area is divided. The number of redundant elements per block e is still one. The general formula, equation (35), is used here, where the number of blocks b is changed in the repair probability term $R_n$ given by equation (31).

## Results and Discussion

This report describes a model that calculates the redundancy yield. The amount of improvement depends on some key factors: the repairable area, available redundant elements, defect density and types of defects and their distribution on the die. The memory area is the area that contributes to redundancy, since the rest of the die area cannot be fixed and has to be functional. Only the random defect density is considered here as the defect category that is potentially repairable. The number of available redundant elements also determines how much improvement can be gained. If there is one redundant element per memory block, only one defect per block can be fixed. The type of defects is another important factor in estimating the redundancy yield. Raster defects such as bits, columns or rows (where bits can represent individual or clustered defects as long as they fall in the same memory block) are considered repairable. Although the number of defects that could be fixed equals the number of redundant elements available, those defects have to follow a certain distribution on the memory array according to the repair probability described in the text.

Figures 2 through 4 show the improvement factor versus the initial yield calculated by the three models: simple, cumulative, and general. Data is compared to three products from two different process generations.

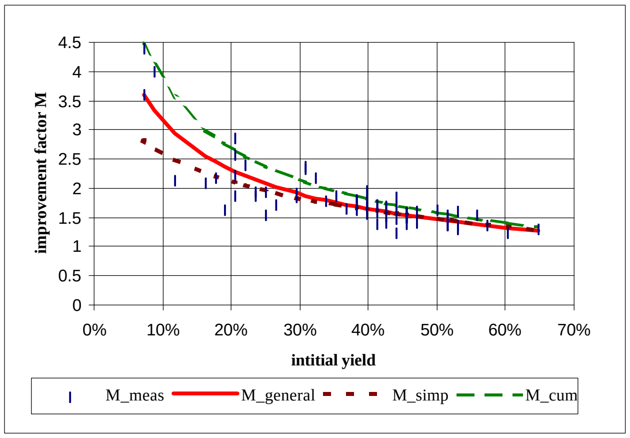

**Figure 2: Three formulas compared to data measured on product 1**

Comparing the formulas of the improvement factor, the closest fit to the actual data was obtained when all the factors affecting redundancy were accounted for (general model). The simple model underestimates the data, since it assumes the repair of die with one defect only, which is not the real case. The cumulative model is overestimating the data. It considers all types of defects and assumes all of them are repairable, if their count is equal to or less than the number of redundant elements. Thus, it ignores the restriction of allowing one defect per block.

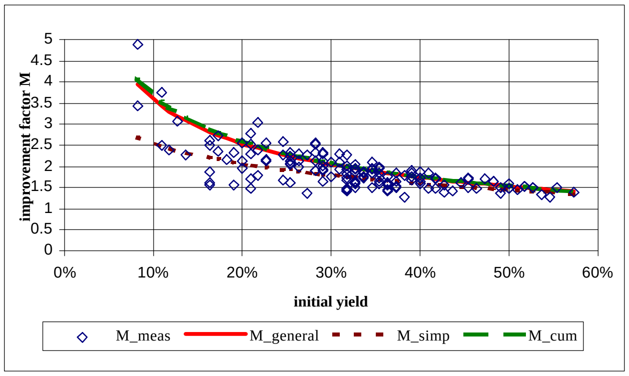

**Figure 3: Three formulas compared to data measured on product 2**

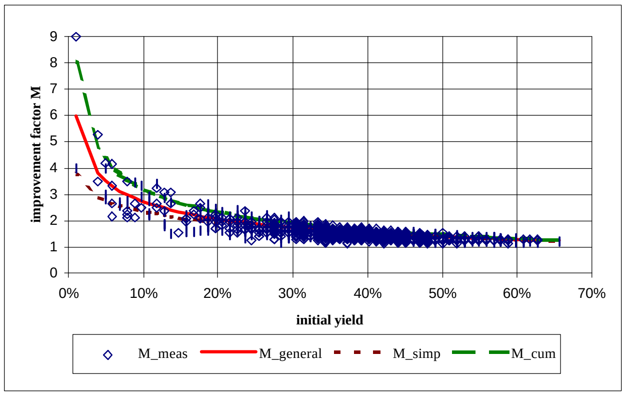

**Figure 4: Three formulas compared to data measured on product 3**

The effect of varying the number of redundant elements is shown in Figure 5. The effect of adding more redundant elements is mostly seen at a lower initial yield. It was observed that the improvement in yield is significant up to two extra sets of elements for a die of originally one redundant element per block. Beyond that, the effect of decreasing the repairable area is dominating, so that the two factors cancel out, and the overall improvement is unchanged.

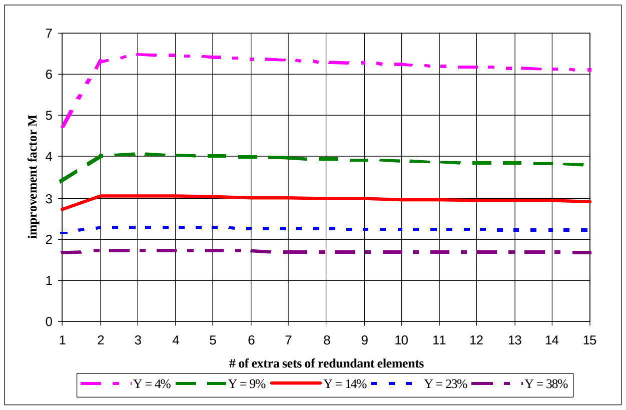

**Figure 5: Improvement factor when extra redundant elements at different initial yield Y are added**

Figure 6 shows the effect of dividing the memory area into a large number of blocks. Again the enhancement in the yield multiplier is observed at a lower yield. With an increase in the initial yield, an improvement in the redundant yield was observed up to six blocks. Beyond that, the effect of more blocks per repairable area is not noticeable, since the higher order terms in the multiplier formula are negligible and do not add extra value to the yield multiplier.

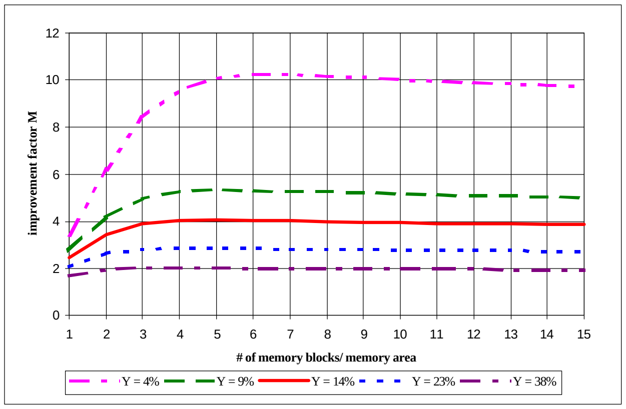

**Figure 6: Improvement factor when the number of memory blocks at different initial yield Y is increased**

## Conclusion

A model for calculating the redundancy yield is developed and described in this paper. Poisson's equation plus the effect of some redundancy-influencing factors are used to derive a general yield multiplier formula. The memory area is considered the only portion of the die area where redundancy is applied. The random defect density is used here as the only defect category that contributes to redundancy. From the defect population, only a fraction of it can be repaired depending on the nature of the defect. According to the die design, the number of repairable defects depends on the available redundant elements per memory block. This means that the number of defects must be below a certain value, and the defects have to follow a certain distribution throughout the memory area to enable redundancy. An efficiency factor is introduced and empirically evaluated to account for the repairable defects. Combining those factors, a general formula is derived and shows good agreement with the actual data. Knowing the properties of a new product and using the efficiency factor for the process generation, the redundancy yield of a new product can be predicted. The formula can also be used to study the impact of varying the number of redundant elements and memory blocks on the final result. Thus, a better design that optimizes the number of redundant elements, memory size with respect to the total die area, and the number of blocks in the memory area might result in a more efficient redundancy scheme.

## Acknowledgments

I would like to thank Dan Grumbling for initiating this project, Tim Deeter for the useful discussions and comments during the development of the model, and Mike Mayberry for his continuous support and guidance throughout this work.

## References

[1] Jitendra Khare, et al. Accurate Estimation of Defect-Related Yield Loss in Reconfigurable VLSI Circuits. *IEEE Journal of Solid State Circuits.* Vol. 28, No. 2, February 1993.

[2] R. M. Warner, Jr. Applying a Composite Model to the IC Yield Problem. *IEEE Journal of Solid State Circuits.* Vol. SC-9, No. 3, June 1974.

[3] B. T. Murphy. Cost Size Optima of Monolithic Integrated Circuits. *Proc. IEEE.* Vol. 52, December 1975.

[4] C. H. Stapper. On Murphy's Yield Integral. *IEEE Trans. Semiconductor Manufacturing*, Vol. 4, November 1991.

## Author's Biography

Nermine Ramadan received a B.Sc. in Nuclear Engineering from the University of Alexandria, Egypt in 1982, and a M. Sc. and Ph.D. in Nuclear Engineering and Engineering Physics from the University of Wisconsin, Madison in 1986 and 1992, respectively. In 1994 she joined Intel in Oregon and is currently working as a Senior Integration Engineer in Sort/Test Technology Development. Her e-mail address is nermine_ramadan@ccm.ra.intel.com

# Redundancy and High-Volume Manufacturing Methods

Christopher W. Hampson, MD6 Cache Product Engineering, Hillsboro, OR, Intel Corp.

Index words: I5, redundancy, raster

## Abstract

This paper will describe practical aspects of a redundancy implementation on a high-volume cache memory product. Topics covered include various aspects of redundancy from a design and product engineering perspective; and present test development methods for future product implementations.

As robust as Intel's wafer fabrication processes are, defects still occur, and wafer yields are the indicator. As die sizes increase, so does the probability of a defective die. Failure analysis has shown that a large percentage of memory array defects are attributed to single-cell defects. This implies that a single memory cell fault can cause an array of over four million cells to be deemed non-functional.

Redundancy is a method wherein "spare" array elements are incorporated into the design to replace elements that have tested defective. First, the basic redundant element strategy must be decided upon. This involves evaluating row, column, and block replacement schemes. Second, the replacement mechanism needs to be a known and reliable entity (e.g., fuses). The design challenge is to select how many redundant elements to add without increasing the die size to the point where the total number of good die is less than the overall yield without redundancy. The critical factors are die size and defect density. Yield forecasts and defect densities for a process are usually available prior to the design phase and are updated on an ongoing basis.

## Introduction

The "I5" is the 512K byte second-level cache packaged with the Pentium® Pro processor. It is one of the first cache devices at Intel to use redundancy. By using this design feature, the I5 has achieved one of the highest yield levels for an Intel product. The overall yield improvement on the I5 with redundancy is a generous 35%, and the cost savings are substantial.

## I5 Architecture

The I5 architecture (Figure 1) consists of seventy-two identical sub-array "blocks" that make up the data array. It is organized into four quadrants, each containing eighteen sub-arrays. One sub-array contains 64K memory cells. Each sub-array corresponds to one input/output bit on the device. "R" represents the redundant elements.

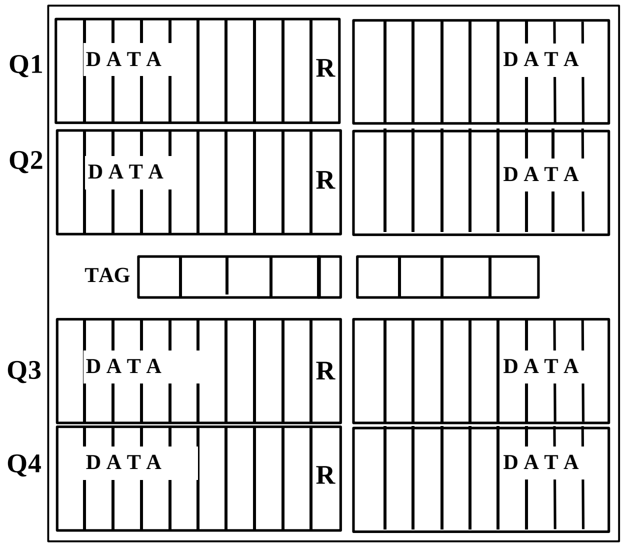

**Figure 1: I5 Basic architecture**

The goal for redundancy involves evaluating several parameters to make a decision on how much redundancy to incorporate. First, the non-redundant yield should be calculated. This is determined from wafer size, number of die, and defect densities for the fabrication process.

For a sample wafer with 33 testable die, and known die size and nominal defect density, the "perfect die" yield might be 20 die per wafer (D/W), without redundancy.

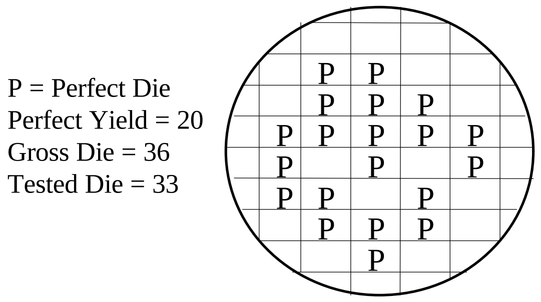

**Figure 2: Non-redundant wafer yield**

With the addition of redundant elements, the die size increases, so fewer die fit on the same size wafer. Hence the "perfect die" yield decreases. We then need to be able to predict for the same defect density, how many additional die can be made functional for a total (perfect plus redundant) die yield.

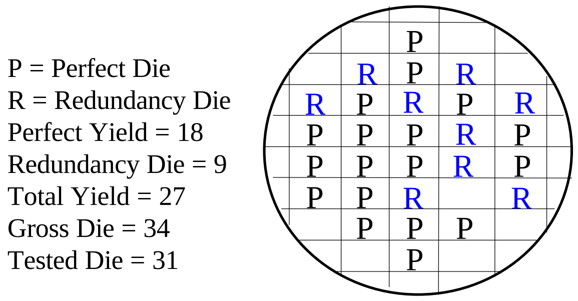

**Figure 3: Total wafer yield with redundancy**

Block replacement was chosen as the optimal strategy for this architecture. Given this block replacement strategy, the yield increase can be determined with defect densities, die size, and sub-array size.

A yield multiplier can be calculated from the equation:

$$\mathbf{MULT} = \mathbf{S}\,(\,1 + 0.01\,(\,\mathbf{N} + \mathbf{I}\,)\;\mathbf{Asb}\;\mathbf{D}\;/\;\mathbf{k}\,)^{\mathbf{k}}$$

| | | |
|---|---|---|
| Where: | **S** | = Programming success rate |
| | **N** | = # of sub-arrays |
| | **I** | = # of redundant sub-arrays |
| | **Asb** | = Sub-array area (mm2) |
| | **D** | = Defect density (#/cm2) |
| | **k** | = Constant for MOS processes |
| | **0.01** | = Conversion from mm2 to cm2 |

**k** is a constant derived from a formula for yield that is based on an average value of the coefficient of variation for the defect density distribution. This yield model is discussed in detail in the paper entitled "Redundancy Yield Model for SRAMS" also published in this issue of the *Intel Technology Journal*.

Since the data array is divided into four quadrants , the logical direction for determining how much redundancy to incorporate in the design was to calculate yield with the multiplier and evaluate for one or more redundant elements (sub-arrays) per quadrant. This process revealed one element per quadrant as the optimum strategy (the sub-arrays labeled "R" in Figure 1). The tag array was evaluated for redundancy and was considered too small for an effective implementation.

Table 1 shows the predicted yield for both the non-redundant and redundant cases. The multiplier equation assumed a programming success rate of 100%. As die size increases due to redundancy, the perfect die yield decreases. For a nominal yield level, the redundant yield multiplier is 1.49 times the perfect die yield of 18, resulting in a 27 D/W final yield. This model predicts at this defect density level, a 50% increase in yield over the perfect die with redundancy; and a 35% increase over the yield without redundancy.

This was the model used to predict yield for the I5. With a nominal defect density, the predicted increase in yield on the I5 was 50% over the perfect die yield with redundancy; and 46% over the non-redundant case.

| One Redundant Element per 18 Sub-Arrays | | | | | | |
|---|---|---|---|---|---|---|
| Yield Level | Defect Density (per cm$^2$) | Perfect D/W Non-Redundant | Perfect D/W with Redundancy | Data Array Yield Multiplier | Total Good with Redundancy | Ratio to Non-redundancy |
| Low | 0.8 | 13 | 12 | 1.85 | 22 | 1.7 |
| Nominal | 0.5 | 20 | 18 | 1.49 | 27 | 1.4 |
| High | 0.2 | 28 | 26 | 1.18 | 30 | 1.1 |
| Non-redundant: Gross Die/Wafer = 36, Redundant: Gross Die/Wafer = 34 | | | | | | |

**Table 1: Sample wafer redundant yield calculation**

## The Replacement Mechanism

The crux of any redundancy implementation is the method used to substitute defective elements with defect-free elements. On the I5, flash cells are programmed to direct muxes that replace the defective sub-arrays with defect-free sub-arrays.

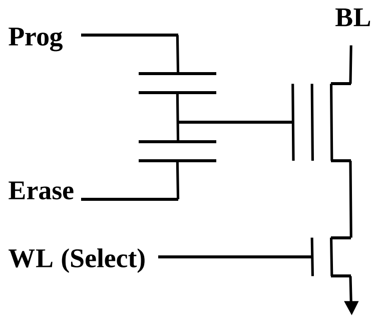

**Figure 4: Flash cell basic schematic**

The flash cell is basically two transistors, one floating gate, and a select gate (Figure 4). To program, a high voltage is applied to the programming gate, and with the select gate turned on, current will flow to the drain of the floating gate transistor. This creates the fields conducive to hot-electron injection, causing an increase in the threshold of the floating gate. Cells should have low unprogrammed switching thresholds (Vt) out of fab, and once programmed, they should have high switching threshold levels. For more information on flash technology, refer to [1].

In Figure 1, a sub-array is replaced by its neighboring sub-array, closest to the redundant sub-array. The sub-arrays between the bad sub-array and the redundant sub-array are also switched to their neighboring sub-array. All this switching is done by muxes in the read and write paths of the device.

One redundant sub-array per quadrant allows for one and only one defective sub-array replacement per quadrant, up to four per die. The task is to determine the number of defective sub-arrays per quadrant. This process is integrated with the cache raster capability. *Raster* is a test process used to uniquely identify all failing cell locations on the device. The redundancy algorithm is integrated with the raster function to identify the failing sub-arrays.

## The Redundancy Algorithm

The basic idea of the redundancy test flow is to find the devices that are defective, evaluate the extent of replaceability (one failing sub-array per quadrant), program the flash cells to effect the replacement of the failing sub-array, and re-test to ensure the redundant sub-array is defect free.

This is accomplished by obtaining fail information from rastering, and modifying tests in the flow, that once executed, will perform the programming and reading of the flash cells. The algorithm is further enhanced by ensuring the cells are programmed to a high reliability level, detecting high Vt cells out of fab, and checks for resorting wafers containing programmed cells.

## The Details

The replacement process occurs early in the test flow, at the Built-in-self-Test (Bist) step. This test checks every memory cell in the data array. The flow starts when this test fails (see Figure 5). This is the only point in the entire test flow at which sub-array replacement is done.

At the raster step, all the failing cell information is collected. It is also discerned if fails occurred in more than one sub-array per quadrant.

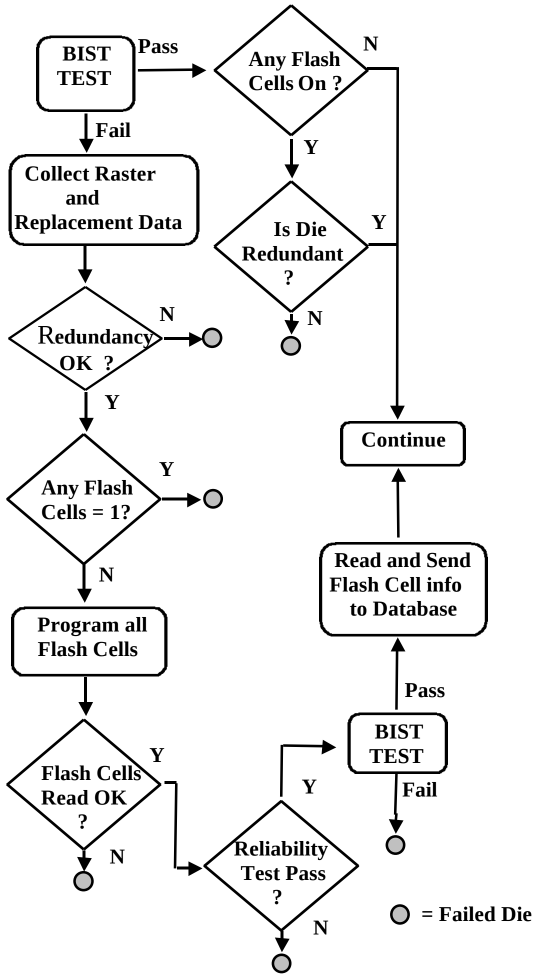

**Figure 5: Redundancy algorithm**

If not, the test flow is halted, and the die is binned non-functional.

First the array is read and tested for all cells equal to "0." This checks for cells whose Vt's are high enough to read as a "1," out of fab. If any cells are read as "1," the die is binned non-functional.

The flash cells are then programmed and tested for the expected contents, and if a cell failed to program, it is binned out. A reliability test is then done to ensure the cells are reliably programmed. This test gives an indication of a high Vt.

The BIST tests are then re-executed, passing die flash cells are read and written to the database for possible future failure analysis, and the die continues the test flow.

If die passes the first BIST tests, the flash cells are read to determine the die's status. If any cells are read as "1," then it must be determined if this is a bad cell out of fab, or a redundancy die. Once this determination is made, the die is either binned non-functional or continues the test flow.

## The Production Results

Raster and replacement data indicated that 85% of all die that failed the BIST screen could use redundancy. After the first month in production, an average increase in yield of 35% was evident. Subsequently, after redundancy had been enabled for two quarters of production, a cost analysis was performed. It showed that all the replacement die had amounted to an equivalent of 6696 wafers. The direct unit cost savings were substantial. In addition to the direct costs, this savings enabled the manufacture and sale of many other Intel products.

## Test Cost of Redundancy

An additional 1.5 seconds was needed to implement redundancy on a die in the sort test program. An analysis was performed to determine if redundancy actually lowered the test time per good die, over an entire lot. Considerations were good and bad die test times with and without redundancy, and time to align wafers and stepping to other die on the same wafer. It was concluded in all cases for different yields that a significant test time savings could be achieved. The actual test time savings at nominal yield levels amounted to 1.33 seconds per good die over the test time without redundancy. Test time savings are greater for lower yields.

## Conclusions

The design yield predictions based on redundancy were somewhat inflated due to the general model used. At nominal yield levels, the predicted increase was (50%); the actual increase was (35%).

First, the factor that would inflate the prediction, yield, is based on the wafer size and therefore is calculated with the gross die per wafer count instead of tested die. This is standard for yield calculations, so the initial predictions counted on more die available for replacement. Furthermore, a major contributor to the degradation of the replacement rate was the 15% fallout for those die whose data arrays had more than one defect per quadrant.

A new redundancy model has been formulated that takes into account the number of "tested die" and the possibility of defect types that do not warrant replacement.

The redundancy application with the I5 has shown that there are other factors that would increase the accuracy of a redundancy model. Quiescent current screening is an important factor that will change for different product types. This screen accounted for an additional 1% reduction in replacement rate, but could be higher for products with tighter testing. The programming success rate seen on the I5 was less than perfect at 97%. This is due to redundant die that had defects in the redundant sub-array, or die that failed to program flash cells. An additional component is the reliability test on the programming element. The position of the replacement function in the test flow, and the test used to determine if a die needed redundancy, are other considerations that can alter the replacement rate. All these factors can be incorporated into redundant yield predictions in the future.

## Summary

Improvements to yield prediction and implementation aspects have been described. The I5 has shown that redundancy makes sense on large arrays, and its benefits are greater for lower yields. It can be implemented and made production worthy, and improved yields and substantial savings can be realized.

## References

[1] Ohsaki, K., Asamoto, N., Takagaki, S., "A Single Poly EEPROM Cell Structure for use in Standard CMOS Processes," IEEE J. Solid State Circuits, vol. 29, No.3, March 1994, pp. 311-316.

## Author's Biography

Christopher Hampson is a product engineer in the Microprocessor Products Group, Cache Products Division. He received a B.Sc. degree in Computer Science from National University, San Diego, Ca. He joined Intel in 1993, was a lead product engineer on the L2 cache for the Pentium Pro® processor, and is

currently working on the next generation of Intel's cache products. His e-mail address is champson@ichips.intel.com

# A PROM Element Based on Salicide Agglomeration of Poly Fuses in a CMOS Logic Process

Mohsen Alavi , Mark Bohr, Jeff Hicks, Martin Denham, Allen Cassens, Dave Douglas, Min-Chun Tsai
Intel Corporation, Portland Technology Development
Hillsboro, OR

## Abstract

A novel programmable element has been developed and evaluated for state of the art CMOS processes. This element is based on agglomeration of the Ti-silicide layer on top of poly fuses. Various aspects of these programmable devices including characterization and optimization of physical and electrical aspects of the element, programming yield, and reliability have been studied. Development of a novel programming and sensing circuit is also included.

## Introduction

The capability of implementing a small PROM array on logic products at no additional process cost is highly desirable for a number of applications such as redundancy implementation in SRAMs, die identification, electrically programmable feature selection, etc.

As CMOS technology scaled, gate oxides became thin enough that implementation of flash memory cells on standard logic CMOS processes (SPEED) became possible [1]. However, further scaling of CMOS technology resulted in inadequate charge retention in the SPEED device due to tunneling of carriers through the gate oxide.

The element presented here avoids the problem with scaled gate oxide thickness. The results are a fuse element which is reliable under thermo-mechanical and bias-temperature stress while enjoying near 100% programming success used in a specially designed circuit. Programming the fuse does not result in any collateral damage in overlying or underlying layers and may be performed nominally at 2.5V and 10 mA in 100 ms.

## General Description of the Element

*A. Physical Properties*

The poly agglomeration fuse (PAF) is made from a polysilicon line shunted on top by a layer of Ti-silicide which is used as the gate in CMOS processes. It is programmed via current stress which results in temperatures high enough to cause agglomeration of the Ti-silicide [2]. The damage due to programming of the element has been found to be very subtle and confined to the Ti-silicide and its interface with the underlying poly layer and the overlying dielectric. The integrity of the entire overlying stack from the passivation to the overlying ILD is found to be intact and no collateral damage has been observed (see Fig. 1,2). This is in contrast with traditional poly or metal fuses which require openings in the overlying layers to facilitate removal of fuse material, and therefore, a post program passivation step. Typically, a fuse link is drawn at minimum allowable width with a few microns of length (see Fig. 2). The effect of fuse doping and geometry on its performance has been investigated extensively and will follow.

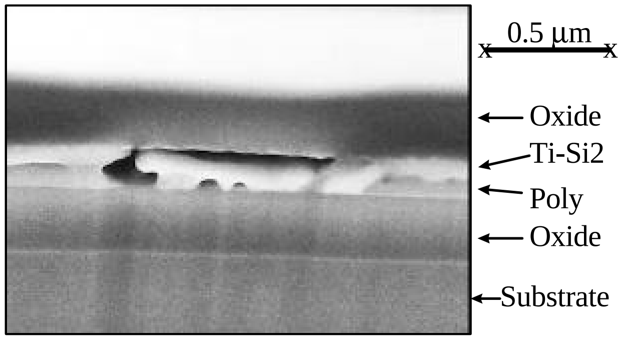

**Fig. 1**, Cross section of the damaged section of a programmed fuse. Lack of collateral damage to the overlying and underlying layers is evident..

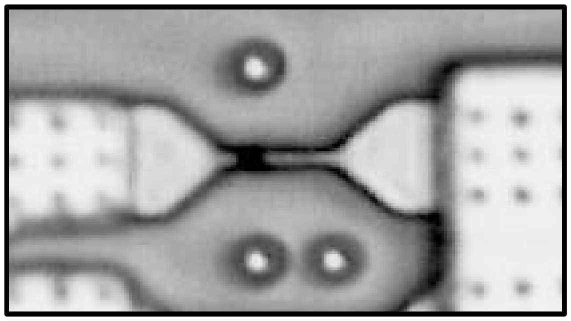

**Fig. 2**, Top view of a programmed fuse. The subtle damage due to programming is evident on the left side of the element.

*B. Electrical Properties*

Prior to programming, electrical properties of the fuse are determined by the salicide layer on top which has a sheet resistance of about 4 ohms per square in our study, resulting in a typical resistance of about 50-100 ohms depending on the dimensions of the fuse. Injection of current beyond a certain level results in a sudden increase in resistance indicating formation of discontinuities in the silicide layer. The value of this resistance varies greatly from device to device. In our structures, post program resistance varied from several hundred Ohms to several hundred kOhms. Post program I-V characteristics are found to be nonlinear and therefore, the value of resistance varies with applied bias.

## Element Characterization

*A. Test Structures*

The element described above has been implemented in a 0.25um CMOS process with a poly thickness of about 0.2um [3]. Ti-silicide films resulting in sheet resistance ranging from 3 to 4 ohms per square have been studied. Initial and post program electrical characteristics of a variety of element designs have been investigated. This includes the effects of poly doping (n, p, undoped), fuse length and width, fuse shape, programming and sensing voltage and current, and programming time.

*B. Programming dynamics*

In order to program an element, a certain amount of current is needed. The voltage needed for injecting this current must obviously be smaller than the available power supply voltage. Under constant voltage stress, as the element gets hot enough, agglomeration starts to occur, thereby, increasing the element resistance. As a result, the current through the element drops to a low value consistent with the elements final resistance and the element cools down. This mechanism is one with negative feedback. Therefore, a given fuse may be stressed only once and it's post program resistance will not increase with additional voltage stress.

Figure 3 shows the I-V characteristics of a typical fuse element. As the voltage is increased, current increases in a nonlinear fashion due to resistance change caused by self heating. When the dissipated power reaches a critical value, fusing occurs and element goes to a much higher resistance.

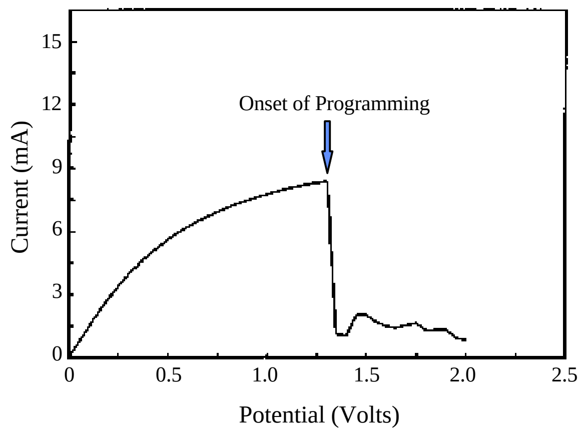

**Figure 3**, I-V characteristics of a typical element upon programming.

*C. Response parameters*

Initial and post program resistance of the element are the two key parameters affecting any circuit meant to sense the state of the element. A maximum value of initial resistance and a minimum value of post program resistance are needed to guarantee proper circuit function (about 100Ω and 1kΩ respectively in our circuit).

Initial fuse resistance depends on element geometry and silicide thickness and quality. Silicide quality in turn depends on process conditions, poly line width, and doping [2,4]. Silicide imperfections are more likely for long narrow elements and best silicide lines were found to be the ones made from p-doped poly. Imperfections in the silicide layer (cracks, high resistance Ti-Si phase) result in a resistive element Figure 4 shows cumulative distribution of the resistance of a typical fuse structure made with two processes with different thermal cycles and Ti thickness. A high resistance tail corresponding to silicide imperfections is evident in the distribution of the resistance of the unoptimized process.

Post program resistance varies greatly from device to device and depends on the shape and size of the discontinuity in the link. Due to this variation, any aspect of this resistance must be studied statistically. Many factors affect the level of fusing and therefore, post program resistance. They include:

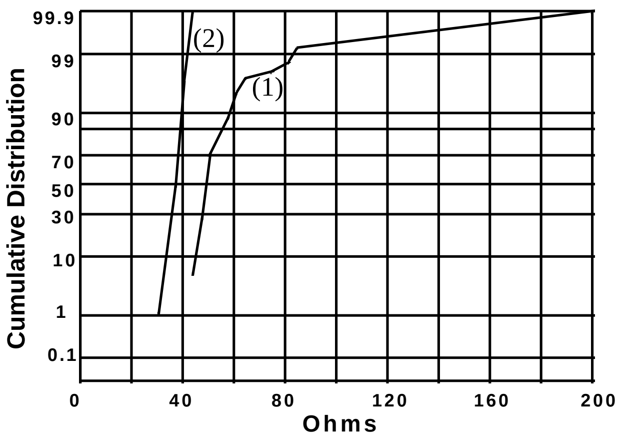

**Figure 4**, Resistance of a typical fuse. (1) Unoptimized silicide process, (2) Optimized silicide process.

*Programming voltage, current, and time:* Even though fusing can occur quickly and at fairly low currents and voltages (in the order of 1V, 8mA, 1mS), post program resistance is significantly enhanced if more energy is dumped into the element Therefore, increased voltage and current levels are needed for a longer time to guarantee a sufficiently large resistance. In this work, minimum programming conditions which resulted in statistically adequate post program resistance were a current of 20mA injected for 100ms with a voltage compliance of 2.5V.

*Initial fuse integrity:* Measured data shows that fuses that are more robust initially (by process or geometry) result in more successfully programmed elements. This is due to the fact that for a given voltage compliance and a given value of fusing current, a smaller resistance results in a larger amount of energy transferred to the device. The fortuitous result is that process conditions which result in good silicide formation and robust unprogrammed fuses also produce elements which program successfully.

*Fuse shape:* In addition to the relation between fuse size and it's initial resistance, the shape of the fuse has a marked effect on the distribution of its post program resistance. This has been found to be due to the fact that in addition to the high temperature necessary for agglomeration, the level of temperature gradient (and therefore stress) in the element plays a key role in the fusing event. Fusing has been found to occur near the sides of the element close to the point which has the highest temperature gradient (see Figure 2,5). Additionally, line width plays a significant role in fusing success with narrower lines having the advantage of better fusing. Figure 6 shows four different fuse shapes of the same length. Figures 7,8 show the distribution of initial and post program resistance for these elements. The difference between post program resistance of elements a,b corresponds to the effect of element width while differences between structures c,d show the effect of temperature gradient.

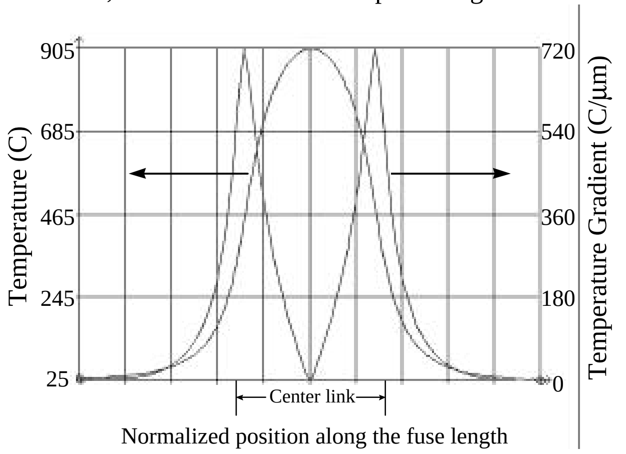

**Figure 5,** Simulation results of profiles of temperature, and temperature gradient along the length of a typical fuse at nominal programming bias.

## Modeling and Simulation

In order to look for an optimum fuse design, numerical simulation of temperature in the element under current stress has been performed. The simulation is based on a two dimensional model with an added loss term to the overlying and underlying layers. Thermal conductivity of the silicide layer and the heat loss coefficient were fitting parameters. Assuming that fusing occurs when the temperature of the fuse reaches 800C (silicide agglomeration temperature [2]), the simulation is able to predict fusing current using a single set of fitting parameters for various fuse geometry with good accuracy (see Fig. 5,9) and provides insight into the distribution of temperature and its gradient in the element.

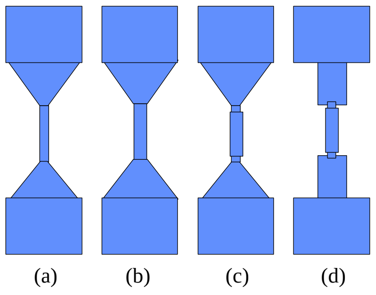

**Figure 6**, various fuse shapes. All elements are p-type, about 2um long. (a) width=0.22um, (b) width=0.27um, (c, d) width = 0.22um/0.27um.

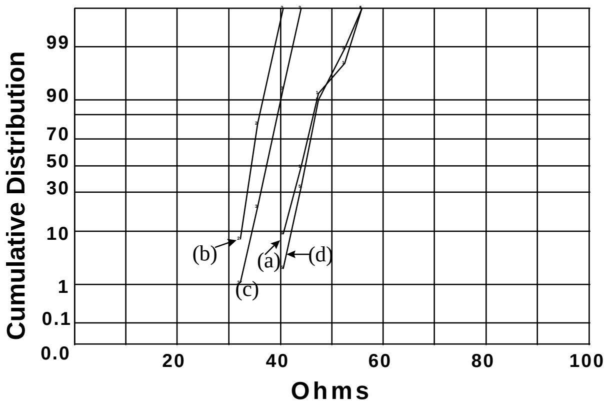

**Figure 7,** Pre-programmed fuse resistance of structures in figure 6.

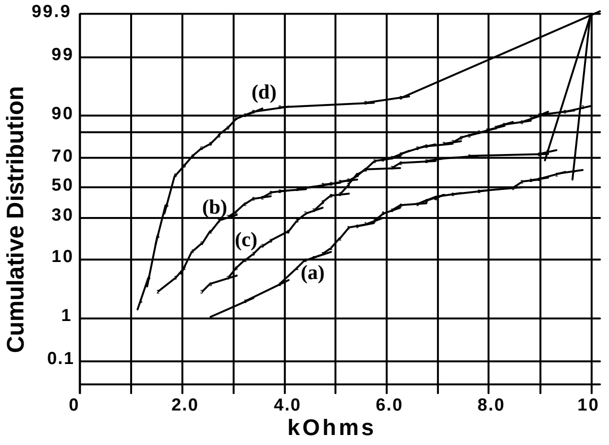

**Figure 8,** Post program resistance of fuses in Figure 6.

## Sensing Circuit

A special circuit has been developed for programming and sensing the element. Figure 10 shows a simplified schematic of this circuit. Programming occurs when a logic LO is asserted on the gate of a large PMOS transistor. Since the fuse programs at relatively low bias, logic and programming circuits share a common supply voltage.

The sensing circuit is a novel and well-balanced solution to a stringent set of requirements, the foremost being that the sensing currents must be kept very low. The core of this circuit comprises a pair of matched N-channel transistors, which perform the sensing, and a pair of matched P-channel devices, which act as current-sensing output loads. The N-channel sensing transistors are connected in a current mirror configuration, such that, if the fuse-reference resistance on the left were equal to the unburned fuse resistance on the right, both circuit branches would have equal current. In practice, the reference resistance is set to about 8 times that of the unburned fuse. This ratio of reference to fuse creates a default (unburned) output voltage that is low enough to be interpreted as a logic LO value. Additionally, for a programmed fuse, the resulting output voltage is sufficiently high to be interpreted as logic HI. Therefore, the gain of the circuit is sufficient for single-ended voltage outputs.

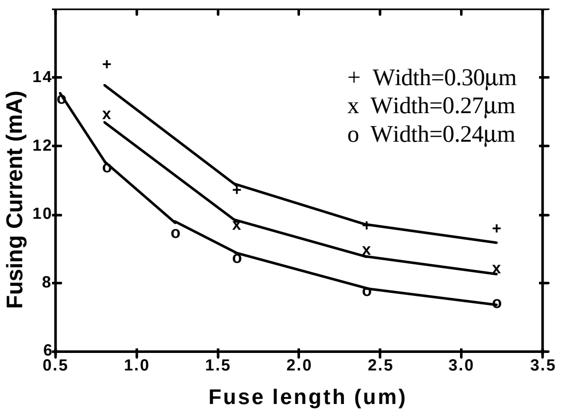

**Figure 9,** Measured and simulated current at the onset of fusing. Solid lines show simulation results. Symbols show measured data points

In this circuit, the ratio of reference to unburned fuse resistance represents a balanced tradeoff between output high voltage (VOH) and output low voltage (VOL) levels. With a ratio of 8, noise margins for VOH and VOL signals are roughly equal. The resulting noise margin is adequate to guard-band the circuit from expected manufacturing variations in transistor Vt and channel length.

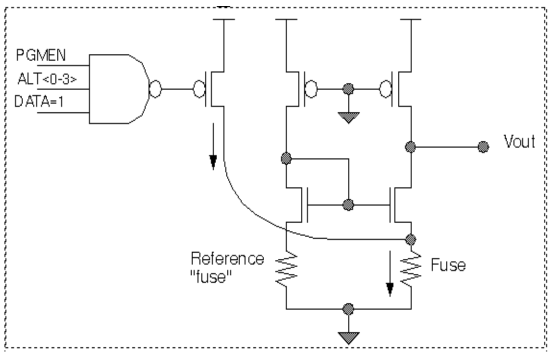

**Figure 10**, Schematic of a simplified programming and sensing circuit.

## Yield and Reliability

A PROM array based on the PAF will suffer yield loss if the programmed fuse does not have a high enough resistance to be properly sensed. Programming yield depends on the fuse design (see Fig. 8), array size, and circuit design. Even after optimizing the element and circuit, the resulting yield may not be as high as expected. In that case, redundant fuse elements are needed such that if programming of a given fuse in a given memory bit is not successful, an additional fuse is available in that bit for an extra attempt. In this work, for a 64 bit array, a programming yield loss of less than 1 in 10,000 was achieved using two fuses per bit (a programmed state in either fuse resulted in a programmed bit).

The reliability of this element was characterized by placing a large number of samples (programmed and unprogrammed) under thermo-mechanical stress (1000 cycles of condition 'C' temperature shock) and in bake (300 hours, 250C). The element was found to be quite stable under these conditions (see Fig. 11). Additional testing was done to characterize the stability of the unprogrammed fuse under bias temperature stress. Results indicated that as long as the sensing current is significantly less than the current at the onset of programming, the device will remain stable.

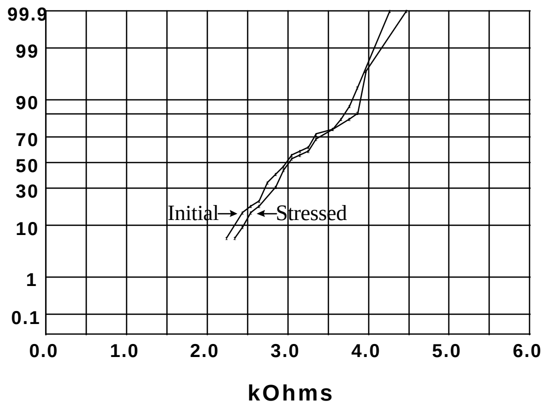

**Figure 11,** Post program fuse resistance distribution of a typical fuse before and after 300HR 250C bake.

## Conclusions

Poly agglomeration fuse is a reliable programmable element which may be implemented in a logic CMOS processes. This element may be programmed under nominal bias and does not introduce any collateral damage. Distribution of the post program resistance depends on silicide quality, fuse shape, doping, and programming conditions. Optimized conditions for fuse shape and programming parameters have been presented using empirical results and numerical simulations and a novel circuit has been presented for the device with a 1 in 10,000 programming yield loss for a 64 bit PROM arrays with 2 fuses per bit. Element reliability has been verified under temperatue shock and bake.

## References

1) K. Ohsaki, N. Asamoto, S. Takagaki, "A Single Poly EEPROM Cell Structure for Use in Standard COMS Processes", *IEEE J. Solid State. Circuits,* Vol 29, No. 3, March 1994, PP. 311-316
2) J.B. Lasky, J.S. Nakos, O.J. Cain, P.J. Geiss, "Comparison of Transformation to Low-Resistivity Phase and Agglomeration of TiSi2 and CoSi2, *IEEE Trans. Elect. Devices,* Vol. 38, No. 2, Feb. 1991, pp. 262-269.
3) M. Bohr, et. al., "A High Performance 0.25µm Logic Technology Optimized for 1.8V Operation", *1996 IEDM Tech Digest,* 1996, pp. 847-850.
4) J.A. Kittle, Q-Z Hong, D.A. Prinslow, G.R. Misum, "A Ti Salicide Process for 0.10 um Gate Length CMOS Technology", *1996 VLSI Symp. Tech. Digest,* 1996, pp. 14,15.