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.

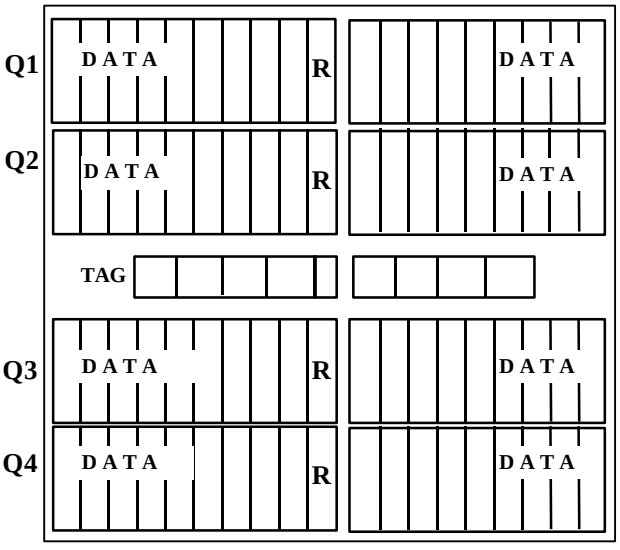

Figure 1: I5 Basic architecture

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.

P = Perfect Die
Perfect Yield = 20
Gross Die = 36
Tested Die = 33

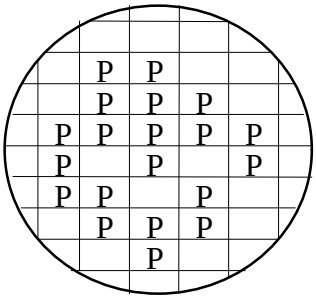

Figure 2: Non-redundant wafer yield

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.

P = Perfect Die
R = Redundancy Die
Perfect Yield = 18
Redundancy Die = 9
Total Yield = 27
Gross Die = 34
Tested Die = 31

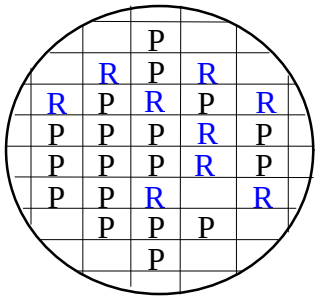

Figure 3: Total wafer yield with redundancy

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:

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

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.

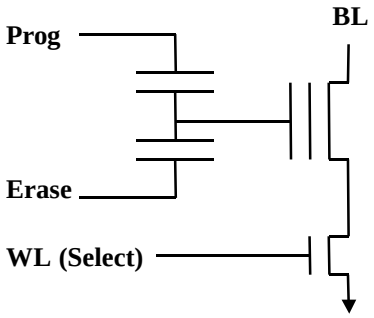

Figure 4: Flash cell basic schematic

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.

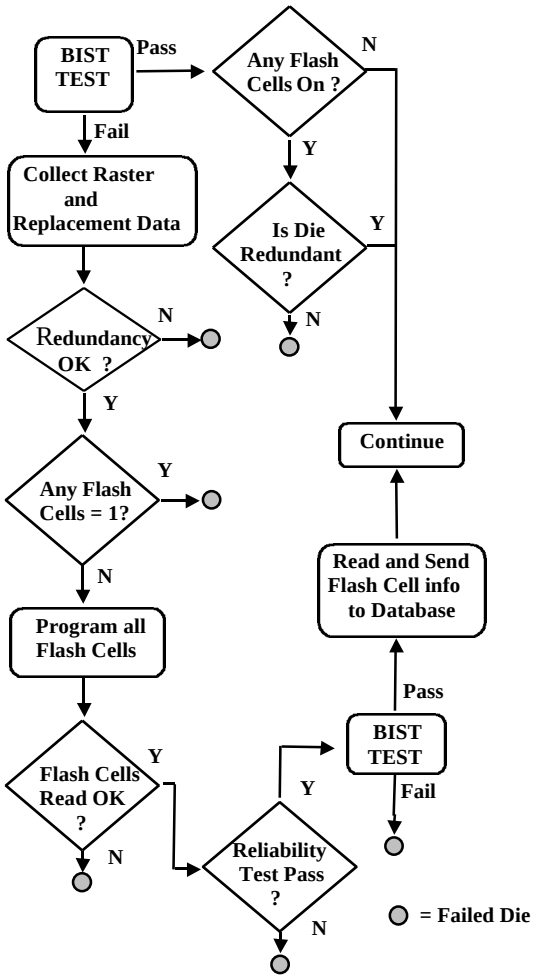

Figure 5: Redundancy algorithm

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