Addressing critical-area analysis and memory redundancy

Simon Favre, Mentor Graphics - March 01, 2012

Design teams—whether using fabless, fablite, or IDM (integrated-device-manufacturer) processes—should handle the goal of reducing a design’s sensitivity to manufacturing issues. The further downstream a design goes, the less likely it is that you can address a manufacturing problem without costly redesign. By promptly addressing DFM (design-for-manufacturing) problems when the design is still in progress, you can avoid manufacturing-ramp-up issues.

One aspect of DFM is determining how sensitive a physical design, or layout, is to random particle defects. The probability of a random particle defect is a function of the spacing of layout features, so tighter spacing increases random defects. Because memories are relatively dense structures, they are inherently more sensitive to random defects, so embedded memories in an SOC design can affect the overall yield of the device.

Understanding how to employ critical-area analysis becomes more important at each successive node. Memories keep getting bigger, and smaller dimensions introduce new defect types. The trade-offs that have worked well on previous nodes may give suboptimal results at the 28-nm node. For example, although manufacturers have avoided the use of row redundancy because they considered it too costly in access time, the technique becomes necessary at the 28-nm node so that vendors can achieve acceptable yields. All of these factors make careful analysis more valuable as a design tool.

Critical-area analysis

Critical area is the area of a layout in which a particle of a given size will cause a functional failure. Critical area depends only on the layout and the range of particle sizes you are simulating. Critical-area analysis calculates values for the expected average number of faults and yield based on the
dimensions and spacing of layout features and the particle size and density distribution that the fab measures. In addition to classic short- and open-circuit calculations, current practice in critical-area analysis includes via and contact failures. Analysis often shows that via and contact failures are the dominant failure mechanisms. You can incorporate other failure mechanisms into the analysis, depending on the defect data the fab provides.

Critical area increases with increasing defect, or particle, size. At the limit, the entire area of the chip is critical for a large-enough defect size. In practice, however, most fabs limit the range of defect sizes that they can simulate, based on the range of defect sizes that they can detect and measure with test chips or metrology equipment.

Defect densities

Semiconductor fabs have various methods for collecting defect-density data. For use with critical-area analysis, the fab must convert the defect-density data into a form compatible with the analysis tool. The most common format is the following simple power equation: \( D(X) = K/X^Q \), where \( K \) is a constant you derive from the density data, \( X \) is the defect size, and \( Q \) is the fall power. The fabs curve-fit the open and short circuits’ defect data for each layer to an equation of this form to support critical-area analysis. In principle, a defect density must be available for every layer and defect type to which you will apply critical-area analysis. In practice, however, layers that have the same process steps, layer thickness, and design rules typically use the same defect-density values.

Manufacturers may also provide defect-density data in a table form that lists each defect’s size and density value. A simplifying assumption is that, beyond the range of defect sizes for which the fab has data, the defect density is zero.

Calculating ANF, yield

To determine the average number of faults for a design, manufacturers use a tool that supports critical-area analysis, such as Mentor Graphics’ Calibre, to extract the critical area for each layer over the range of defect sizes. To achieve this goal, manufacturers measure the layout and determine all of the areas in which a particle of a given size could result in a failure. The tool then uses numerical integration, along with the defect’s size and density data, to calculate the expected average number of faults, according to the following equation:

\[
ANF = \int_{D_{\text{MIN}}}^{D_{\text{MAX}}} CA(X) \times D(X) DX,
\]

where \( ANF \) is the average number of faults; \( D_{\text{MIN}} \) and \( D_{\text{MAX}} \) are the minimum and the maximum defect sizes, respectively, according to the defect data available for that layer; and \( CA(X) \) and \( D(X) \) are the critical area and the defect-density data, respectively.

Once the manufacturer has calculated the average number of faults, it is usually desirable to apply one or more yield models to make a prediction of the defect-limited yield of a design. The defect-limited yield cannot account for parametric-yield issues, so be careful when attempting to correlate this figure to actual die yields. One of the simplest and most common yield models is the Poisson model: \( Y = E^{-ANF} \), where \( Y \) is the yield, \( E \) is a constant, and \( ANF \) is the average number of faults. It is generally simpler to calculate the average number of faults and the yield for cut layers, such as contacts and vias, than for other layers. Most foundries define a probabilistic failure rate for all single vias in the design and
assume that via arrays do not fail. This simplifying assumption ignores the fact that a large enough particle can cause multiple failures, but it greatly simplifies the calculation of the average number of faults and reduces the amount of data the fab must provide. The designer needs only a sum of all the single cuts on a layer and can calculate the average number of faults as the product of the count and the failure rate.

**Memory redundancy**

Embedded memories can account for significantly large yield loss in SOCs due to random defects. Although SOCs can use other types of memories, assume that the design uses embedded SRAM. Typically, SRAM-IP (intellectual-property) providers make redundancy an option that designers can choose. The most common form of redundancy is the use of redundant rows and columns. Redundant columns are typically easier to apply because they address only the multiplexing of bit lines and I/O ports—not the address decoding.

To analyze failures with critical-area analysis, it is important to define which layers and defect types are associated with which memory-failure modes. By examining the layout of a typical sixor eight-transistor SRAM bit cell, you can make some simple associations. For example, by looking at the connections of the word lines and the bit lines to the bit cell, you can associate diffusion and contact to diffusion on column lines with column failures. Because contacts to diffusion and contacts to poly both connect to Metal 1, row and column layers must share the Metal 1 layer. Most of the layers in the memory design find use in multiple places, so not all defects on these layers will cause failures that are associated with repair resources. Irreparable, or fatal, defects, such as short circuits between power and ground, also occur.

**Repair resources**

Embedded-SRAM designs typically use either built-in self-repair or fuse structures that allow multiplexing out the failed structures and replacing them with the redundant structures. Regardless of the method of applying the repair, the use of redundant structures in the design adds area, which directly increases the cost of manufacturing the design. Additional test time also increases cost, and designers may have a poor basis for calculating that cost. The goal of analyzing memory redundancy with critical-area analysis is to maximize defect-limited yield and minimize the effect on die area and test time.

A critical-area-analysis tool can accurately analyze memory redundancy only if it knows the repair resources available in each memory block, the breakdown of the failure modes by layer and defect type, and which repair resource these failure modes are associated with. Calibre can specify these variables as a series of critical-area-analysis rules. Each memory block also requires a count of total and redundant rows and columns. To identify the areas of the memory that can be repaired, you can either specify the bit-cell name that each memory block uses or use a marker layer in the layout database to allow the tool to identify the core areas of the memory.
Listing 1 provides the sramConfig memory-redundancy specification. The first two lines list the critical-area-analysis rules—that is, the type of defects that can occur—that have redundant resources for a family of memory blocks. The first two lines also contain the column rules and the row rules. These rules depend on the type and the structure of the memory block but are independent of the number of rows and columns and the redundancy resources. The last two lines describe an SRAM block design and specify, in order, the block name, the rule-configuration name, the total columns, the redundant columns, the total rows, the redundant rows, the dummy columns, the dummy rows, and the name of the bit cell. In this example, both block specifications refer to the same rule configuration, sramConfig. Given these parameters, Calibre calculates the unrepairable yield using the defect-density data that the fab provides.

**Yield with redundancy**

Once the critical-area-analysis tool has performed the initial analysis, providing the average number of faults without redundancy, you can calculate the yield with redundancy. Calibre uses a calculation method employing the principle of the Bernoulli Trials, according to the following equation:

\[ Y_R = \sum_{K=0}^{N_F} \binom{N_F}{N_R-K} P^{(N_F-K)} Q^K, \]

where \( N_F \) is the number of functional, nonredundant memory units; \( N_R \) is the number of redundant memory units; \( P \) is the probability of success, or yield, you derive from the average number of faults; \( Q \) is the probability of a failure \((1 - P)\); and \( \binom{N_F}{N_R-K} \) is the binomial coefficient, which is a standard mathematical function. If the critical-area-analysis tool can postprocess the calculations with different memory-redundancy specifications, it can present numeric and graphical output that makes it easy to visually determine the optimal amount of redundancy. The goal is to ensure the required number of good units out of a total number of units.

To see how effective memory redundancy can be, consider a hypothetical example. The memory of interest is a 4-Mbit, 32-kbit×128-bit SRAM. The goal is to realize at least 128 good units from a total of 130 units. In this case, there are two units needing repair and no defective units. Analysis determines that the unit yield considering one defect type is 0.999. The unrepairable yield of the entire core is then 0.999 raised to the 128th power, or 0.8798. If you perform the analysis for all defect types, the expected yield is approximately 0.35.

If you add redundancy to repair any unit defects, the repaired overall yield is 0.999. Memory designers use the repair-ratio metric to express the efficacy of memory redundancy. It stipulates that the repaired yield minus the unrepairable yield divided by one minus the unrepairable yield equals the repair ratio. A value in the high 90s is good. In this case, the repair ratio is \((0.99 - 0.35)/(1 - 0.35)\), or 0.985.
Using Calibre to determine an optimum redundancy configuration, you must first set up a configuration file for the tool (Listing 2). The bit-cell name, ram6t, tells the tool the name of the hierarchical-layout element that describes a memory unit that can be repaired and that you should consider in this analysis. This name enables the tool to calculate the critical area of the entire memory core, including all instantiations of ram6t.

Listing 2 Configuration Entries for 4-Mbit SRAM

```
sramConfig =
{ [DIPF.OPEN] {DIPF.SHORT} {single.OHDC0} {M1.SHORT}
{M1.OPEN} {single.VIA1} {M2.SHORT 0.6}
{M2.OPEN 0.6} 
{PO.SHORT} {PO.OPEN} {single.POCO} {M1.SHORT}
{M1.OPEN} {single.VIA1} {M2.SHORT 0.4}
{M2.OPEN 0.4} {single.VIA2} {M3.SHORT}
{M3.OPEN} }
sram_2048x32_core sramConfig 34 2 2050 2 0 0 ram6t
sram_128x32_core sramConfig 34 2 130 2 0 0 ram6t
```

With this configuration information, Calibre calculates the average number of faults for memory with no redundancy, as well as for various redundancy configurations. Figure 1 shows the results as a table with values of the average number of faults for different redundancy configurations. The table rows show the results for the entire design, for just the memory, and for specific types of defects. In the highlighted row, the average number of faults for the 1024×32-bit memory core improves substantially; the failure rate in Column 6 is half that in Column 5. To achieve this improvement, Column 6 includes one redundant row, but adding a second redundant row shows almost no further improvement (Column 7).
Figure 2 lists the effects of redundancy schemes in terms of repair ratio by design total, by total of all analysis layers, by memory, by block, and by layers or groups. Figure 3 shows a tool-created plot depicting the average number of faults for each redundancy configuration and for each type of defect. The combination of one redundant row and one redundant column causes a large decrease in the average number of faults, and adding resources has little further effect. From these results, you can deduce that the expected average number of faults is based on the layout of the memory under consideration and the defect density of the fab and the process. The designer can now determine the effect of various redundancy configurations on the expected yield of an embedded memory.
Memory redundancy is intended to reduce manufacturing cost by improving die yield. If no redundancy is applied, alternative methods to improve die yield may include making the design smaller or reducing defect rates. If you apply redundancy to parts of the design in which it has no benefit, then you waste die area and test time, increasing manufacturing cost. Between these two extremes, you apply redundancy or not, depending on broad guidelines. Designs with high defect rates may require more redundancy; those with low defect rates may require no redundancy. The analysis of memory redundancy using critical-area analysis and accurate foundry-defect statistics is necessary for quantifying the yield improvement and determining the optimal configuration.

Acknowledgment
A version of this article originally appeared on EDN’s sister site, EDA Designline.

Author’s biography
Simon Favre is a technical-marketing engineer in the Mentor Graphics Calibre division, where he supports and directs improvements to the Calibre Yield Analyzer product. Before joining Mentor Graphics, Favre worked for Ponte Solutions, which Mentor acquired in 2008. He previously worked at other EDA companies, as well as at several semiconductor companies. Favre has bachelor’s and master’s degrees in electrical engineering and computer science from the University of California—Berkeley.