CHALMERS

Integrated Circuit Yield Enhancement

Redundant Multi Core DSP Cluster

Master of Science Thesis

in Integrated Electronic System Design

MIKAEL ANDERSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, August 2010

Integrated Circuit Yield Enhancement
Redundant Multi Core DSP Cluster

Mikael L. Andersson

Examiner: Lars Svensson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden August 2010

Abstract

The manufacturing of integrated circuits is not a perfect fault-free process. The constant downscaling of integrated circuits requiring higher accuracy each generation also allows the designer to fit more transistors in the same area. From a manufacturing point of view, this downscaling introduces additional possible sources of error, which forces a constant struggle to keep the yield or ratio of successfully manufactured chips high enough to be profitable. The manufacturing success ratio, or *yield*, is a major component of what determines the time and material cost it takes to manufacture an integrated circuit.

To increase yield a common approach is to add redundant or spare parts to the system requiring only enough of them to work.

Redundancy has long been a common concept in memories. For random logic blocks, the overhead cost has been too large to be reasonable. But for multi core systems, with several instances of the same logic block, the situation is starting to resemble the case of a memory.

The manufacturing yield can be predicted by statistical models. Such models may consist of analytical expressions based on very low level information such as the exact layout features of every transistor and wire of the entire chip. However the addition of a spare core for a multi core cluster is an architectural decision that has to be made in the early stages of the design where low level details are not readily available.

The modeling approach taken in this project uses a collection of simplified models previously used for yield calculations for memories. The model results in estimates of the yield before and after the addition of redundant cores. The input parameters to the model are based on gate counts and global routing estimates of an early floor plan together with information about the manufacturing process in the form a fault density.

When sources of defects are found and suppressed, the yield ramps up. The yield model presented here also shows the effect the redundancy has on the yield ramp, pushing it towards an earlier volume production date, potentially decreasing the product time to market.

Acknowledgement

I would like to thank my supervisor at Chalmers, Lars Svensson and my supervisors at Ericsson AB, Edvin Catovic and Per Ingelhag.

I thank you for sharing with me your expert knowledge, theoretical insight, great suggestions, feedback and everything that made this project possible.

Also I would like to thank Ulf Gmoser and staff at the ASIC & Advanced FPGA department, and everyone else at the Lindholmen office site.

I thank you all for your constant guidance and support throughout this entire project!

This thesis work was carried out for Ericsson AB, at their site Lindholmen, Gothenburg Sweden during the spring 2010.

About Ericsson:

Ericsson is a world-leading provider of telecommunications equipment and related services to mobile and fixed network operators globally. Over 1000 networks in more than 175 countries utilize Ericsson's network equipment and 40% of all mobile calls are made through their systems. Ericsson is one of the few companies worldwide that can offer end-to-end solutions for all major mobile communication standards.

Abbreviations

ASIC	Application Specific Integrated Circuit
BIST	Built-In Self Test
CMP	Chemical Mechanical Polishing
DFM	Design for Manufacturability
DFY	Design for Yield
DSP	Digital Signal Processor
GCPW	Good Chips per Wafer
GDPW	Good Dice per Wafer
IC	Integrated Circuit
IRR	Internal Rate of Return
NoC	Network on Chip
NPV	Net Present Value
SPC	Statistical Process Control
SoC	System on Chip
STA	Static Timing Analysis
SSTA	Statistical Static Timing Analysis

1 Introduction

1.1 Background

Application specific integrated circuits (ASIC) consist of nanometer-sized components which are intuitively very difficult to manufacture.

The ratio of the number of circuit chips that pass the post manufacturing tests over the total amount of manufactured chips is what is referred to as *manufacturing yield*.

$$\text{yield} = \frac{\text{chips manufactured successfully}}{\text{chips manufactured in total}} \quad (1)$$

Yield is a major component in calculating ASIC manufacturing cost. Yield prediction is therefore an important issue. The standard method to model yield is by attempting to separate mechanisms contributing to yield loss. Manufactured chips are either discarded because they fail the functional tests or the non-functional tests, such as speed and power tests.

The most common source of failure in the functional tests is random defects, caused by particles interfering with the manufacturing procedure. The discarded chips are said to be in a category usually referred to as defect limited yield, or simply *defect yield loss*.

The most common source of failure in the non-functional tests is the limited accuracy in the manufacturing procedure, which causes parameter variations in different layers of the design. These chips are categorized under *parametric yield loss*, but are usually divided further into *process systematic yield loss* and *design systematic yield loss*. The former is caused by for example lithography misalignment, or uneven manufacturing steps. The magnitude of the process systematic yield loss is reduced as the manufacturing process matures over time.

Design systematic yield loss causes chip designs that follow the specified design rules still to be difficult to manufacture. This yield loss category is expected to dominate in the lower nanometer era [1], but little documentation regarding this subject has been found.

The effects of both defects and parametric yield loss are increasing with the constant downscaling of feature sizes. At each new technology node, small defects previously considered harmless are becoming a threat, and previously acceptable accuracy for varying performance is no longer enough. The manufacturing process therefore has to continuously improve to keep yield at a reasonable level.

Another way of trying to improve yield is to design more robust systems, less susceptible to defects and parametric variations. The engineering fields striving to suppress yield loss from a design point of view are referred to as Design for Manufacturability (DFM) or Design for Yield (DFY).

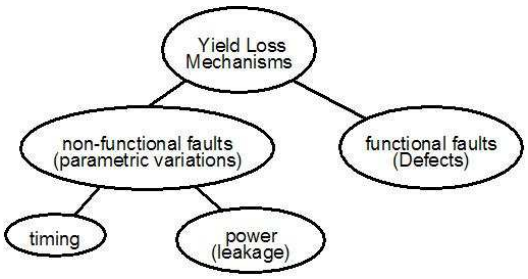


Figure 1: Conceptual illustration of the characterization of yield loss mechanisms

1.2 Purpose

The purpose of this project is to examine the possibility of improving ASIC manufacturing cost from a design perspective, assuming minimum insight into a specific manufacturing process implementation.

Currently, for a chip to pass the post fabrication tests, all circuit blocks needs to be functionally correct and remain so at least above or below some limit in both timing and power analysis.

Recent trends, however, show chip performance scaling through the use of multiple functionally identical logic units working in parallel. This trend may allow a reduction in the manufacturing test acceptance level by introducing redundant copies of system components. The redundant copies may replace faulty originals, and thereby increase the probability that the manufactured chip fulfils the system requirements. If the yield can be modeled and predicted, then it will be possible to derive the number of redundant units required to reach the desired yield target.

The added redundant units will, however, increase the area needed per chip. This in turn might increase the raw yield loss as well as increase the overall fabrication cost for the chip design.

In summary, both low yield and a large chip area affect the ASIC fabrication cost in a negative way. Tradeoffs between the two will have to be examined. Is it possible to find a sweet spot where the overall cost is at a minimum?

1.3 Goals and subtasks

1. Develop a yield model:

To be able to examine the effects of redundancy on the yield of a system on chip (SoC) design, a model estimating the yield has to be developed first. Subjects and tasks that need to be examined are the following:

- What are the main contributions to yield loss?
- What statistical theories are needed to model the different yield loss mechanisms behavior?
- How early in the design can the yield model be used? Can the model provide useful results with only a netlist of standard cells?
- Can the model details be hidden from a user at a higher abstraction layer? An example could be to estimate the effects of routing for a given net list, before a particular routing tool has been used.

2. Expand the yield model to account for redundancy.

Assuming the yield model developed is accurately predicting the ratio of chips per wafer that have been manufactured with all components functional; how can the model be expanded to account for having only a sufficient number of instances of functional subsystems pass the test, instead of all of the instances passing the test? An example would be if a chip has 2 MB of memory, but only 1 MB have to be functionally correct to fulfill the system specification.

3. Improving yield through redundancy on block level.

Apply the yield model to a case study to evaluate overall trends of redundancy added to a system.

4. Find a way to express the optimal amount of redundancy.

Increasing the chip area of a design reduces the number of chips per wafer, thus decreasing the parallelism in the manufacturing steps, and in the end increasing the cost per unit.

Increasing the yield of a design increases the ratio of useful dice per wafer, so in the end reduces the cost per unit.

When does it pay off using redundancy and to what extent?

The procedure for testing chips with redundancy might increase in difficulty or time spent. At least an additional configuration step should be expected to determine which sub-components should be used to represent logical topology of the system.

2 Problem description – Productivity

The main underlying goal of the project is to examine possibilities to increase the potential profit for a System on Chip (SoC) based product. Many variables affect the profit of a product, for example time to market, gross margin, and labor and material cost. The efficiency of the silicon foundry is hard to influence, and this report will focus on reducing the manufacturing cost per chip from the ASIC designer's point of view. But what knobs can be turned to reducing the overall cost?

It is well known that many manufacturers set a price on a chip design based on its size: the larger the design, the higher the price [2]. There are two main reasons behind this.

First, the integrated circuits are produced not one-by-one, but instead several chips are fabricated in parallel on a thin circular silicon wafer to increase the throughput. The manufacturers can only fit a certain number of chips on the wafer at one time, so the larger the chip design is, the fewer copies can fit on the wafer, and so fewer copies can be manufactured simultaneously. The price is affected since the number of wafers required to produce a certain number of units increases as well as the time required to produce them.

> **Example 1.**
> Assume an ASIC-design company which has two chip designs D_1 and D_2 and wants to produce 1000 of each product at a manufacturer with an advertised 100 % yield. D_1 is twice the size of chip design D_2.
> If 100 instances of D_1 fit per wafer, then 10 wafers are needed to reach the desired volume.
>
> If the number of chips that fit on a wafer is assumed to decrease proportionally as a linear function of growing chip size, then D_2 should be able to fit 200 chips on a wafer and so only 5 wafers are needed to reach the desired volume of 1000 units.
>
> If the manufacturing company sets the price per wafer then it will cost less to produce 1000 units of design D_2 than D_1.

The example above shown above is a great simplification and a more accurate relation between chip size and the number of chips that fit on a wafer will be examined in more detail in section 3.

The second reason that price increases with a larger area is that the many manufacturing process steps are not perfect, and typically several chips per wafer are not successfully manufactured. The results of the successfully manufactured chips over the total number of chips attempted can be monitored and translated into a probability. The probability that a circuit element is manufactured correctly is commonly referred to as the *empirical yield model* of that circuit element.

The manufacturer assumes a certain worst case circuit density. For a fixed circuit density, the larger a chip area is, the more circuit elements potentially fits into the area, and must be manufactured correctly for the chip to pass the functional post-fabrication tests.

Commonly, the yield for a chip with *N* circuit elements is modeled as the product of the yield for each individual circuit element. The chip yield is a probability function and thereby also defined to be between zero and unity, so the yield monotonically decreases as the number of required circuit elements per chip increase (see section 5.1). The chip yield thereby limits the number of chips that will be manufactured in parallel.

Example 2.
Assume the company has developed two additional chip designs D_3 and D_4 of equal area. Again 1000 units should be produced. The manufacturer this time has a yield model predicting D_3 to have a chip yield 20% and D_4 with a chip yield of 40%. Looking at the definition of yield once more and inputting the desired target of 1000 units to be manufactured successfully and the yield predictions to estimate the number of chips that need to be produced and thereby also see how much time and material would be spent.

$$\text{yield} = \frac{\text{manufactured successfully}}{\text{manufactured volume}}$$

$$\{\text{manufactured successfully} = \text{desired target}\} \Rightarrow$$

$$\text{manufactured volume} = \frac{\text{desired target}}{\text{yield}}$$

Then D_3 requires 5000 chips on average to be manufactured to compensate for the yield loss while D_4 requires 2500 to reach 1000 functional units.

A wafer can therefore be seen as both as a material cost and a time cost. A difference between how chip yield and chip size affect the number of good chips per wafer is that after manufacturing the chips have to be tested thoroughly. The size of a chip does not affect the number of chips that go through the testing stage, and thereby indirectly does not increase the time spent on testing. But for a lower yield, the number of chips that will go through the test stage to reach target number of units increases; the testing stage is where the defective chips are detected.

The discussion above motivates an ASIC designer to strive for a small chip area to minimize manufacturing cost. This report will examine more complex statistical expression for determining yield in the next few sections, and show that a low yield is not always the result of a large total chip area.

More generally, the productivity problem could be described as maximizing the number of sellable chips per wafer [3,33].

In the formula below, the yield factor does not include the packaging process, when relatively small chip input and output pins are connected to the larger pins used to connect the chip to the outside world. This processing step can also be said to have a separate yield factor of its own, yet the chip designer has very little influence on the success rate. The packaging yield will not be covered in this report.

To sum up what has been covered above:

$$\text{number of good chips per wafer} = \text{chip yield} \cdot \text{potential chips per wafer} \qquad (2)$$

Yield determines the probability of manufacturing a “good” chip.

Chip area determines the number of “potential chips per wafer”.

The number of chips that fits on a wafer is a deterministic function of the chip area and will be covered in the following section.

3 *Potential number of chips per wafer*

The first part of the productivity formula shown in section 2 deals with answering the question: How many chips fit on a wafer?

The basic intuitive formula for estimating the number of chips, with area *A*, that fits on a wafer is the ratio of the wafer's circular area and the chip's rectangular area:

$$\text{number of chips per wafer estimate} = \frac{\pi R_{eff}^2}{A} = \frac{\text{usable circular wafer area}}{\text{rectangular die area}} \quad (3)$$

where R_{eff} represents the effective wafer radius which is the total wafer radius with the outer edge of the wafer subtracted because it is known to have higher sensitivity to failure

$$R_{eff} = R_{tot} - R_{edge} \quad (4)$$

The chip area *A* is formed by (chip width + *s*)·(chip height + *s*) where the s variable is determined by the precision of the separation process, dicing the wafer into individual chips.

de Vries has observed [4] many similar attempts to improve the above estimate and has categorized approaches as improving accuracy either through multiplying by a correction factor or through subtracting a correction term.

Some formulas take aspect ratio of the chip area into consideration, and some do not, depending when the estimate is intended to be used.

$$\text{aspect ratio} = \frac{\max\{\text{width}, \text{height}\}}{\min\{\text{width}, \text{height}\}} \quad (5)$$

Earlier in the product development stage only a rough estimate of the chip area might be achievable. Examples of estimate formulas [4]:

$$\text{number of chips per wafer estimate} = \frac{\pi R_{eff}^2}{A} \text{correction factor} = \frac{\pi R_{eff}^2}{A} e^{-\sqrt{A}/R_{eff}} \quad (6)$$

$$\text{number of chips per wafer estimate} = \frac{\pi R_{eff}^2}{A} - \text{correction term} = \frac{\pi R_{eff}^2}{A} - cR_{eff}/\sqrt{A} \quad (7)$$

However, the above formulas (6) and (7) are not quite accurate since each chip is a discrete instance, and only chips that are fully within the effective wafer area can be counted towards the total. The formulas also fail to distinguish two half chips on the edge from one complete chip, so it will overestimate the number of chips that fit on a wafer.

To get a more accurate estimate, a small detail must be considered. Because the wafer will be diced, cutting the wafer into separate chips, a regular grid is preferred to reduce the area needed for the separation process, so all rows and columns must be aligned, indicating that either all rows contain an even number of chips or all rows contain an odd number of chips and not a mix of both.

The problem could be viewed as a grid with equal row and column spacing corresponding to the chip dimensions, and trying to fit a circular area on top, cutting as few chip corners as possible.

Numerical algorithm

A numerical algorithm was presented by de Vries [4] to calculate the exact count of chips per wafer, given the wafer radius and chip dimensions.

The algorithm can be summarized in the following steps:

1. Start with a fixed number of chips in a row. Then assume this same row is pushed up until the outmost chips are as close to the edge of the wafer as possible.

2. Calculate how many dice that can fit in a row "above" the current row. This is repeated until the combined row height of adding another row, from the center, would reach outside the wafer edge.

3. Calculate how many dice that can fit in a row "below" the current row. And this is further divided into two cases when the outermost chip is limited by the

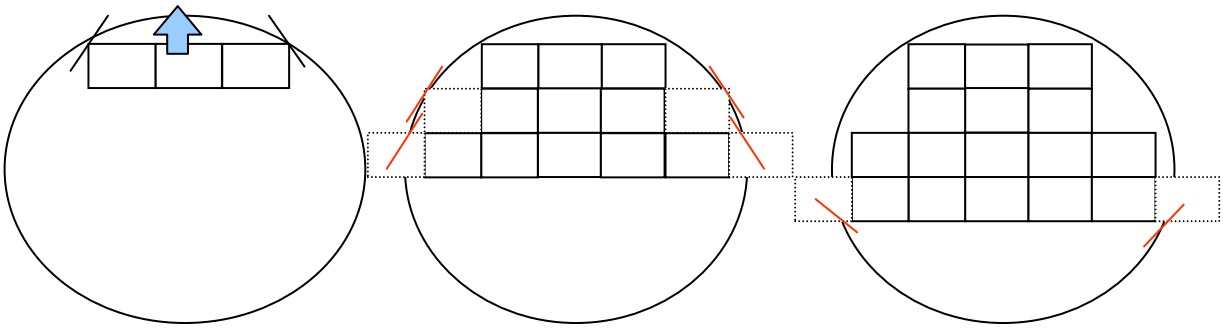
A. top corners

B. bottom corners

Step 3 is repeated until the combined row height from the center would reach outside the wafer edge.

Steps 1 to 3 are repeated for all reasonable number of dice in the starting row, from 1 to when the width of the start row would exceed the wafer diameter.

The maximum result among all the alternatives is then returned as the estimate for the number of chips per wafer.



a) b) c)

Figure 2: Illustrations of the steps in the numerical algorithm. a) *Step 1*, pushing the row of chips towards the top of the wafer, until the top corners hit the wafer edge. b) *Step 2*, adding more chips until the top corners cut the wafer depicted by the red lines. c) *Step 3*, same as b) except for the lower half wafer the chip's bottom corners are checked to be inside the wafer edge

Overestimating the number of good chips per wafer will lead to an unexpected decrease in potential profit, either when it is discovered that the number of sellable products are less than expected if a fixed number of wafers was purchased, or when the number of wafers required to reach a target volume is higher than expected, leading to a higher cost.

The main focus of the next section will be on predicting the functional yield, the other factor of the "number good chips per wafer" formula (2).

4 Yield prediction model

In section 2, about productivity, it was mentioned how the manufacturer had a model to predict the yield of a chip design. It is a very powerful tool to estimate the expenses in material and time cost. This section will focus on different yield loss mechanisms and on how they have been modeled.

The two main contributors causing a loss in manufacturing yield are *defect generated faults* and *parametric variation*.

4.1 Random defects

The yield losses caused by defects originate from the raw materials that might include impurities which interfere with the manufacturing process. Another source of defects is from the outside environment: dust particles or gas clouds from the process equipment or the human operators. Defects are the main reason why the IC manufacturing takes place in so called clean-rooms where the air is filtered to minimize the number of defect generated faults.

Defects can cause unwanted “bridges” between conductive lines: short circuits which cause nodes in the circuits to be permanently connected to either the voltage supply or to ground. These faults are known as stuck-at-one or stuck-at-zero, respectively. Defects can also cause open circuits because of missing conductive material, and can leave nodes floating [5].

Both opens and short circuits can be caught because they cause incorrect outputs. During functional testing, the circuit is fed with input vectors and produces output vectors. These are compared to a known desired result; any deviation is considered a functional fault and is a strong indicator that a defect has occurred.

Some defects might not directly alter the output of the circuit. An example is an inverter with a resistive short between the voltage source to the output [5]. The resistance of the short is too high to pull the output to a logical high value at an expected low output, but instead causes the circuit to draw orders-of-magnitude higher current by causing a resistive path directly from the voltage source to ground.

4.2 Defect limited yield modeling

Not every defect causes a fault in the manufacturing process. If a dust particle lands in an area of the chip that is basically empty or free of circuitry, there is no harm done. The area in which a defect will lead to a fault occurring is usually referred to as a circuit's sensitive or critical area.

According to Stapper [6], a random defect model consists of two main components. The first part is the statistical distribution of the number of faults across the area in question. There have been many different opinions about the appropriate distribution to use. The second part is a parameter for the statistical distribution, but needs careful definition of its own: the average number of faults.

4.2.1 The first part: statistical distribution function

Poisson model:

A common method to model faults caused by defects is to use a Poisson-based random variable. The Poisson distribution is derived from the Binomial distribution taken to a limiting case. Let X represents the Poisson random variable taking on different possible values x denoting the number of faults on the integrated circuit during manufacturing.

$$\Pr(X = x) = \frac{e^{-\lambda}\lambda^{x}}{x!} \qquad (8)$$

In a chip yield prediction formula there is only one interesting special case of (8), when the chip in question is fault-free. The Poisson expression for the case when the circuit is fault-free, or equivalently the probability of zero faults, is then described as follows

$$\text{Chip yield} = \text{probability of zero fault of type } i = \Pr(X = 0) = e^{-\lambda_i} \qquad (9)$$

The variable λ, called the average number of events, is both the expected value and variance for a Poisson random variable. The very same variable contains information about the specific case for describing its usage, and so it needs to be scaled accordingly; but more on that in section 4.2.2.

The Poisson distribution is also commonly used to model reliability R as

$$R(t) = P_s(t) = \Pr(X = 0) = e^{-\lambda t} \qquad (10)$$

where P_s is the probability of success. Notice however the time dependence, which does not apply for manufacturing yield modeling, since any defective units are disposed of before they are put into use.

Negative binomial model:

The most commonly used statistical distribution for modeling defect based yield seems to be the negative binomial model. Stapper describes the negative binomial model to be the result of many approaches such as starting from a binomial model [6]. Below in (11) is the general expression of the negative binomial model.

$$\Pr(X = x) = \frac{\Gamma(\alpha + x)}{x!\,\Gamma(\alpha)} \frac{(\lambda/\alpha)^{x}}{(1 + \lambda/\alpha)^{\alpha + x}} \qquad (11)$$

Again, the computation of defect based yield deals with the case of zero events or faults, which greatly simplifies the expression just as in the case of the Poisson expression. The

following equation (12) describes the negative binomial model expression for the probability of zero faults.

$$\Pr(X = 0) = \frac{1}{(1+\lambda / \alpha)^{\alpha}} \qquad (12)$$

The negative binomial model takes defect clustering into account with the coefficient α (alpha). With an increase in the α coefficient, less clustering of defects is assumed by the model; as α goes towards infinity, the model reduces to the Poisson model, which assumes no clustering at all. Figure 3 shows the Negative Binomial model with a fixed value for the average number of faults, while varying the clustering coefficient.

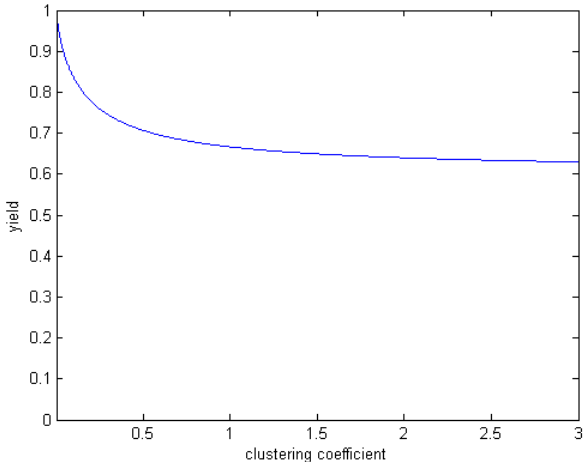


Figure 3: Negative Binomial distribution function for a fixed λ = 0.5 and a varying α.

Correlation between process steps can be taken into consideration with the negative binomial distribution to produce a yield model, which computes the probability of zero faults on a chip. As shown in [6], correlated process step will be treated mathematically as though they were independent.

The negative binomial model was in a subsequent article by Stapper [7] shown to be poorly scaling with circuit area when the feature sizes were reduced in the technology nodes during the early 1990s. Designs with the same chip area were found to have widely different yield. The simple yield formula used λ, the average number of faults, which was obtained from thorough analysis of one design in a particular technology node, figuring that it would be a decent representative of the process, and then rescaled by areas:

$$\lambda_{new} = \lambda_{old} \frac{A_{new}}{A_{old}} \qquad (13)$$

Stapper found that a better fit was acquired if the yield model was scaled based on the number of circuits, assuming a different model parameters are used for each different type of circuit elements such as DRAM, SRAM, and random logic. This scaling seem intuitively correct when considering for example random logic standard cells where an area is increased to twice the size is not always the case that it will contain twice the number of standard cells as the interconnect gets more complex and needs more room.

Both the Poisson and Negative binomial model are derived from an approach where Murphy showed a yield model, developed during the 1960s, described by a Poisson random variable compounded with a distribution function of the average defect density variable [8]. The compounding function seemed to have served the purposed of describing the fluctuation

across the wafer, from wafer to wafer or from lot to lot. Murphy's yield model described this compounded yield model as

$$Y = \Pr(X = 0) = \int_0^\infty e^{-Da} f(D) dD \quad (14)$$

where a represented the critical area and D was the defect density.

Murphy showed three different potential candidates for the f(D) function; the Dirac or δ-function, a triangular function as a simple estimate of the Gaussian distribution function, and lastly a uniform distribution.

Later it was shown by several researchers that a really good fit of the model to the test data was observed if the compounding function was a gamma distribution. The resulting yield formula was then the negative binomial model (12) described in the beginning of the section.

4.2.2 The second part: the average number of faults

In the case of modeling defect generated faults, the variable λ (lambda) describes the average number of faults per chip or circuit. How can λ be broken down into defect generated faults?

Recall how the Binomial random variable is the origin of the Poisson random variable [9]:

$$\Pr(X = x) = \binom{n}{x} p^x (1-p)^{n-x} \quad (15)$$

The Binomial variable itself is described as a number of n statistically independent Bernoulli trials with probability p for one of the two possible mutually exclusive outcomes of the Bernoulli trial. The variable λ is the expected value of the Binomial distribution, which is the product of n and p [37]. Then conversion to describe the average number of faults can be seen as simply letting:

$$\lambda = E[X] = n \cdot p \quad (16)$$

where

X: the number of faults
n: number of defects
p: the probability that a defect becomes a fault

The number of defects n is generally described per unit and is scaled to fit the specific problem instance. A general model for computing the average number of faults is:

$$n = D \cdot U \quad (17)$$

In (17) the D represents the defect density per unit and U for the number of units in question.

Each different fault type might need a different way to represent and model them.

The most common forms would be the calculation of the average number of short circuits between conductive lines, including cross layer shorts or so called pinholes, and open circuit faults. In this case, it seems like the most common approach is to let the D represent average defect density per area and U represents A, the total circuit area.

To derive the p variable, it is commonly implicitly assumed that inside an independent circuit area the defect locations have a uniform distribution or rather an equal chance of occurring across the chip's surface. The probability that a defect becomes a fault differs depending on defect size, and is represented as a function θ, commonly known as *probability of fault* given a certain defect of size x:

$$p = \theta = \Pr(\text{defect becoming a fault} \mid \text{defect}) = \frac{A_{cr}}{A} \quad (18)$$

An example of an independent circuit area is a cluster as assumed by the Negative Binomial distribution. For the Poisson distribution, the whole chip can be seen as an independent area.

Because of how both n and p depends on the total area, A can be cancelled out of the equations

$$\lambda = E[X] = n \cdot p = D \cdot A \cdot \frac{A_{cr}}{A} = D \cdot A_{cr} \quad (19)$$

and so defect density is equivalently scaled with the A_{cr} function representing the circuit's critical area.

4.2.3 Critical Area

The critical area is generally defined as the area of a circuit design's layout that will cease to function correctly if a defect would occur on it [10]. Defects can have many shapes and forms, but for computational simplicity it is commonly assumed that a defect is to be represented by a circular area with a diameter x. The location of the defect is represented by the coordinates of the centre of the circular shape.

The critical area for a break off or an open conductive line as an example is the area where the defect of size x can land on a wire to reduce the resulting effective width below a predetermined critical minimum width value w_{min}, when the resistance of the conductive material would simply be too high to be considered functionally correct.

Let $w_{min} = 0$ for simplicity; then, for a defect with a diameter equal to the wire width w, the defect centre has to be positioned perfectly in the middle of the wire to cause an open fault, however the defect can be located anywhere along the wire length L. The chance of a defect occurring there and therefore the critical area is defined as zero for defects sizes less than the wire width.

$$A_{open}(x) = 0 \qquad x < w \quad (20)$$

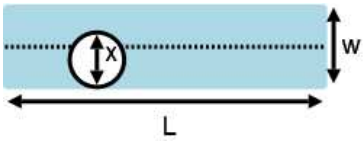


Figure 4: conceptual illustration of a blue conductive wire with a defect of harmless size to the left and a harmful defect size to the right

If the diameter of defect is instead larger than the wire width, the defect has some leeway. When the defect is positioned across the wire, Stapper [11] observed that

$$A_{open}(x) = \mathrm{L}(x - w) \qquad x \geq w \quad (21)$$

An illustration of (21) can be seen in Figure 5.

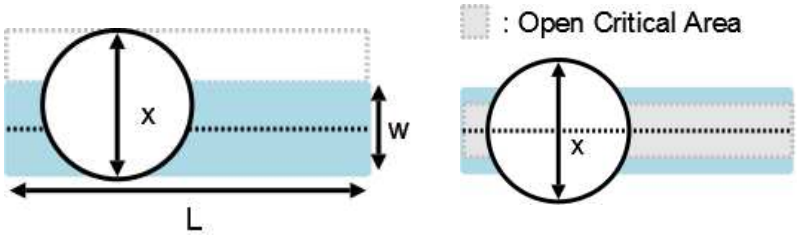


Figure 5: a conductive wire and a harmful defect size, shows how the difference between diameter and width converts into critical area

The critical area of a bridging or short between two conductive lines can be expressed in a similar way, by considering the break of the isolating space instead. The width of the spacing is denoted *s*. However the length of the spacing is the length of the area where the two wires run in parallel.

The critical area of a circuit is defined to be less than the total area of the circuit in question. When the theoretical critical area grows larger, the critical area remains capped at the maximum total area even for larger defect sizes. The mathematical expressions get a bit more complicated since placement of the wire on the chip needs to be taken into consideration; since a wire displaced from the middle will have its associated critical area stop growing at one side but still continue at a reduced growth rate at the other side.

For more than one wire, a similar approach can be taken. The critical area is simply multiplied with the number of parallel wires.

$$A_{open}(x) = N \cdot \mathrm{L}(x - w) \qquad x \geq w \tag{22}$$

However at a certain defect size the area between the wires join together as a large area, and only continues to increase not as N areas but as one wire, with a constant factor of all the wires and spaces between. Again Stapper [11] formulated the following expression

$$A_{open}(x) = \mathrm{L}\left(2\left(\frac{x}{2} - w\right) + Nw + (N\text{-}1)s\right) \qquad w < x \leq 2w+s \tag{23}$$

$$A_{open}(x) = \mathrm{L}(x + (\mathrm{N} - 2)w + (N\text{-}1)s)$$

More complex structures require more elaborate equations, and research in critical area analysis has focused on deriving the critical area for larger designs. Fortunately straight lines are difficult enough to manufacture that at sub-100-nm technology nodes, there are recommended design rules for layout, stating that all wires on a layer should go in the same direction, and every layer alternates between horizontal and vertical as the preferred routing direction. So to make an L shape in layout you would have to change layer to go in the other direction. For 45 nm or below, these recommended rules have become mandatory rules for at least the poly layer [12,13]. Although it is unfortunate that straight lines themselves are difficult to manufacture, it should simplify the critical area analysis to the point when parallel wires are the only cases left to consider for open and short circuits caused by defects.

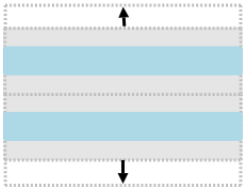


Figure 6: multiple parallel conductive wires. Shows how the critical areas have grown together and continues to expand only from the top and bottom most wires.

4.2.4 Defect density

Defect density, just as critical area, also depends on defect size *x*. It is described by the so called defect size distribution function, which describes the probability of a certain defect size occurring together with $\overline{D}$, the average number of observed defects per area unit regardless of size.

$$D(x) = \overline{D} \cdot pdf_{defect}(x) = \overline{D} \cdot dsd(x) \qquad x_{\min} \le x \le x_{\max} \tag{24}$$

The function is truncated for defect sizes below a certain value; defects smaller than $x_{\min}$ will not harm wherever it may occur. Defect sizes above $x_{\max}$ have such miniscule probability of occurring that it is rounded down to zero for simplicity.

The defect density per area reflects the manufacturing process cleanliness. With each new step scaling down feature sizes, the defect density have to constantly strive to improve. The previously considered harmless defect sizes will now cause opens and shorts. Also the previously harmful defect sizes now cover a larger critical area of the circuit and thereby have a larger probability of becoming a fault. The aim of the manufacturer is to have at least as low average number of faults as the previous process node, and this requires a constantly cleaner manufacturing process [12].

The defect density has to be determined empirically by manufacturing test structures. The test structures are tested and faults are recorded. Techniques exist to map the test results of the Built-In-Self-Test (BIST) from real designs or test chips to individual fault types on different layers [14]. Then indirectly it can be determined that the faults must have occurred in an associated critical area.

Thus, given fault test data and the critical area for the test circuit layout, the defect density can be determined for every defect size *x* at a desired resolution.

$$D(x) = \lambda(x) / A_{cr}(x) \tag{25}$$

The defect densities determined with these techniques are then reused to predict yield in other chip designs.

If empirical testing is out of the question, there are models that describe the defect density based on data fitting of past measurements.

Defect size distribution general form is modeled for example by Stapper [11]:

$$D(x) = \begin{cases} \dfrac{2(n-1)\overline{D}x}{(n+1)x_0^2} & ,0 \le x \le x_0 \\ \dfrac{2(n-1)\overline{D}x_0^{n-1}}{(n+1)x^n} & ,x_0 \le x \le \infty \end{cases} \tag{26}$$

Here $\overline{D}$ describes the average defect density or the average number of defects per area unit. The variable *n* is process dependent; empirical results in the past indicate that *n* = 3 give results that match observed test data very well [6,11].

$$D(x) = \begin{cases} \dfrac{\overline{D}x}{x_0^2} & ,0 \le x \le x_0 \\ \dfrac{\overline{D}x_0^2}{x^3} & ,x_0 \le x \le \infty \end{cases} \tag{27}$$

The variable x_0 is the peak value of the distribution or in other words the most probable defect size. A typical defect size distribution function is plotted in Figure 7.

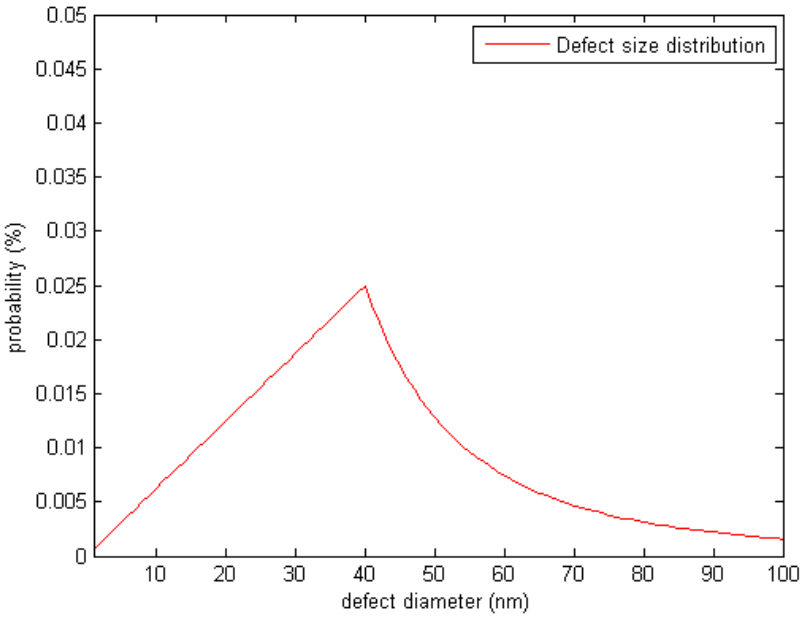


Figure 7: Non-truncated defect size distribution.

4.2.5 Combining critical area and defect density

Both the critical area and the defect density depend directly on defect size; and to obtain the total probability of a fault in the circuit in question, it is necessary to sum the effect of all defect sizes.

The average number of faults or the expected number of faults can be calculated to be

λ = Average number of faults =

= Average number of defects $\cdot$ Probability that a defect become a fault=

= {Average number of defects = Average number of defects per unit area $\cdot$ Area in question}

$= \overline{\overline{D}A}\sum_{\forall x} \Pr(\text{defect become a fault} \mid \text{defect size x}) \cdot \Pr(\text{defects of size x}) =$

$= \overline{D}A\sum_{\forall x} \Pr(\text{defect located in critical area} \mid \text{defect size x}) \cdot \text{defect size distribution(x)} =$

$= \overline{D}A\sum_{\forall x} \Pr(\text{defect located in critical area} \mid \text{defect size x}) \cdot \text{defect size distribution(x)} =$

= {assuming locally uniform distribution of defects over the area in question} =

$= \overline{D}A\sum_{\forall x} \int_{a \in A_{cr}(x)} \frac{1}{A} ds \cdot f_D(x) = \overline{D}A\sum_{\forall x} \frac{A_{cr}(x)}{A} \cdot f_D(x) =$ {Infinite number of sizes x}=

$$= \overline{D}\mathrm{A}\frac{1}{A}\int_{\forall x} A_{cr}(x) \cdot f_D(x) dx \qquad (28)$$

From here there are two main alternatives to represent λ:

Alternative 1

$$\lambda = \overline{D}\int_{\forall x} A_{cr}(x) \cdot f_D(x) dx = \int_{\forall x} A_{cr}(x) \cdot \mathrm{D}(x) dx \qquad (29)$$

The alternative should be the most commonly used together with critical area analysis tools. These tools aim to extract the critical area directly from the layout of the design for all possible defect sizes.

Alternative 2

$$\lambda = \overline{\mathrm{D}}\mathrm{A}\overline{\theta} \qquad (30)$$

where $\overline{\theta}$ represents the average probability of failure over all possible defect sizes.

Expression (30) is the one used by Stapper who worked closely with IBM and is therefore also commonly seen in IBM articles related to yield prediction [6]. There was much secrecy in the 1970s involving methods for indirectly determining critical area which is a computationally very heavy task. Later, the principles were published in scientific articles [15]. The main idea was the use of computer simulations to estimate θ, the probability that a defect becomes a fault, by placing defects at random across the layout and analyze if any damage was caused. The process was repeated for all known defect sizes until a desired resolution had been achieved.

The second alternative representation of λ (30) can mislead the user into believing that lambda scales with the circuit area; but this is not the case as seen in the previous steps. The total area which houses the circuitry is irrelevant, and the important thing to remember is that the equation only scales with the critical area.

For predicting the yield of new products early in the design stage, when the final layout is not yet available and thereby also any critical area analysis is out of the question, it is useful to assume a constant $\overline{\theta}$ and make your early economic assessment based on a scaled area.

Using *Alternative 2* and simplifying the expression further by letting $A\bar{\theta}$ denoting an average critical area $\overline{A_{cr}}$ results in a commonly seen expression:

$$\lambda = \overline{D} \cdot \overline{A_{cr}} = DA \qquad (31)$$

This expression looks like a simplification of *Alternative 1* of expressing the average number of faults, just as in the beginning of this chapter when defect size was not an issue. Note again that care must be taken to not misinterpret what the variables represent and how the expression is meant to be used.

Multiple fault types

Fault types are plenty and can have different origins, and each type would need a separate λ to accurately describe them. The principle for deriving critical area is identical for all layers, but what differs is the feature sizes of the layout such as the line width and spacing between lines. Also, the defect distribution is modeled differently depending on the process steps involved with each layer. Each has different minimum-maximum range of interesting defect sizes as well as the shape of the defect size distribution, all depending on the materials used in the process steps.

The most commonly seen approach to combine the faults of all the different layers associated with a particular area is to simply sum them together [6].

$$\lambda_{tot} = \lambda_{L1\_open} + \lambda_{L1\_short} + \lambda_{L1\text{-}2\_short} + \lambda_{via1\text{-}2\_open}\ \lambda_{L2\_open} + \lambda_{L2\_short} + \lambda_{L2\text{-}3\_short} + \lambda_{via2\text{-}3\_open}\ \lambda_{L3\_open} + \\ + \lambda_{L3\_short} + \lambda_{L3\text{-}4\_short} + \lambda_{via3\text{-}4\_open} \ldots \qquad (32)$$

That concludes the defect yield modeling theories that are needed to understand the approach taken to scale up the yield model and to later include redundancy.

4.3 Parametric yield loss – static power and frequency

Unlike defects which cause the circuit design to divert from its intended functional behavior, parametric yield loss is referring to chips that fail to live up to the non-functional requirements. These chips are considered not to be manufactured correctly because they do not meet the performance requirements of for example speed and power consumption.

The name "parametric yield loss" refers to the parametric variations that are a natural part of the manufacturing process. Examples of the parameters in question that tend to vary are the thickness, width, and height of the layout features. These parameters are known to vary and the manufacturing process itself is characterized by so called three sigma window commonly used in statistical process control (SPC) [16]. A parameter for example a width *w* is then modeled as a random variable

$$w = E[w] \pm 3\sigma \qquad (33)$$

where σ (sigma) is the standard deviation. For a normal or Gaussian distribution, the random variable has a 99.73% probability to be within the three sigma window.

Example 3.
A circuit's critical path is determined to be Gaussian and has a standard deviation of 5 ns. Assume a simplified case where the only requirement is that the final product needs a clock period no greater than 65 ns or:

$$t_{\text{clock max}} \leq 65\text{ns}$$

Setting the design goal to be 65 ns would without regarding the standard deviation then 50% of the manufactured product to be above the mean. If instead subtracting a safety margin of 3 σ to the requirement

$$t_{\text{design max}} \leq 65\text{ ns} - 3\sigma = 65\text{ ns} - 3 \cdot 5\text{ ns} = 50 \leq 65\text{ns}$$

This then results in a probability of less than 0.135% that the manufactured circuit has a period above 65 ns.

Unlike defect limited yield loss, which has been documented since the 1960s when the basic circuit elements (the transistors) were manufactured one per chip [8], the parametric yield loss has not received as much attention over the years. But over 50 years of constant scaling down the feature sizes of the transistors to keep up with Moore's law [17], the parametric variations have gone from variations happening across wafer batches called lot-to-lot, to wafer-to-wafer variations, to across the wafer or chip-to-chip variations and finally to intra-chip variations [20].

The traditional methods for designing circuits have been to ensure that the worst case performance is still adequate, for example using static timing analysis (STA) [18]. However, those methods assume deterministic parameters, and then it is enough to consider only one case that is the bottleneck of the system. When timing is modeled to have a statistical distribution, then *every* path, even the designed shortest path, has a non-zero probability to be the bottleneck and must be considered. *Statistical* static timing analysis (SSTA) has evolved from STA to model non-deterministic gate delays.

Theoretically, the sum of several uncorrelated Gaussian random variables is known to be Gaussian as well [9]. Paths might share gates leading to their delays to be correlated increasing the complexity. The maximum value of Gaussians is known to not have a Gaussian distribution, estimating it is a more difficult problem [18].

When considering power analysis, things are a slightly simpler since only the sum of the power consumption needs to be considered, and there are no alternative paths. A logic block

on the chip with unusually high power consumption might be acceptable if it occurs with a block that has lower power consumption. Analytical models for static power consumption have been reported to exist for gate level calculations [19].

Unfortunately, the complex nature of both statistical power and timing analysis makes it impossible to do more than just scratch the surface of the subjects in a short project such as this. There are books with several chapters that cover the variation of each manufacturing process step that causes the variation of one transistor [20].

A really good overview on the subject of parametric yield modeling is given by Agarwal et al [21]. The authors show how the parameters that vary during manufacturing actually affect both timing and power.

Power consumption consists of both dynamic and leakage power. The critical dimension or the minimum feature size of an IC is the transistor's effective gate length L_{eff}. The gate delay of an IC is proportional to $1/L_{\text{eff}}^2$. The dynamic power consumption also known as the switching power is linearly dependent on capacitance and the capacitance is linearly dependent on L_{eff}. When dynamic power is the dominant power component the downscaling of

$$\text{case I}: \text{dynamic} >> \text{leakage}$$

$$L_{eff} \downarrow \Rightarrow \text{power} \downarrow \text{AND delay} \downarrow$$

$$\text{caseII}: \text{leakage} >> \text{dynamic}$$

$$L_{eff} \downarrow \Rightarrow \text{power} \uparrow \text{AND delay} \downarrow$$

A statistical yield model would in the end look somewhat like

$$\text{Parametric Yield} = \Pr\left(\sum_{\forall i} \text{Power consumption}_i < P_{\max} \cap \max_{\forall i}\{\text{delay}_i\} < t_{\max}\right) \qquad (34)$$

No further attempts to develop a parametric yield model were made in this project, and in the analysis in the rest of the report, the parametric yield will only be represented by a constant yield factor Y_P. The parametric yield variations were handled in a similar fashion in the past. All the unknown yield loss mechanisms that had yet to be modeled were collected in a single yield factor Y_0, also known as the gross yield factor (e.g. in [6]).

The parametric yield loss is reported [1,19]to become the dominant yield loss mechanism in processes below 100 nm even though no real numbers dividing yield loss into parametric yield loss compared to defect limited yield have been found.

4.4 Block yield

The statistical distribution and the average number of faults explained in the previous sections are used to represent the manufacturing yield for the area in question. The area itself does not have to be the whole chip at once, and can instead represent partial blocks or sub-chips. The yield expression for these blocks can then be reused if the blocks occur in another design, for example an adder, shifter or cache. For defect generated yield, the following would be an expression for a single block:

$$Y_{block} = \frac{1}{\left(1 + \lambda_{block} / \alpha\right)^{\alpha}} \quad (35)$$

The traditional way of combining multiple yield source factors have been to simply multiply the yield factors [6]:

$$Y_{block} = Y_0 Y_{defects} Y_{\substack{parametric \\ variations}} \quad (36)$$

where the yield terms correspond to the gross yield, the defect limited yield and the parametric variation limited yield respectively. The defects and parametric variation limited yield have been discussed above, but the gross yield represents all the other miscellaneous failure types that either have no associated yield model or unknown causes. The unknown gross yield factor is usually represented by a constant, to fit the model prediction closer to the observed test results once it is known.

The same approach for combining yield source factors has also been used for combining block yield factors into expressions for bigger designs.

$$Y_{chip} = Y_{DSP} Y_{MEM} Y_{func1} Y_{func2} Y_{func3} = Y_0 \prod_{\forall i} Y_{block\,i} \quad (37)$$

There are more complex variations of block yield models that include correlations between the blocks [22, 23], but the yield model expressions become unnecessarily large with a constant representing the statistical dependence between every pair of blocks. Also, statistical tests for correlation requires large amount of test data for each pair of blocks and since no test data is available in this project that would be a large number of options that could potentially do more harm than good.

Therefore, the correlation considerations between blocks have been left out, but is a part of the yield model that could be improved if the predicted yield would end up completely off target. The assumption of independent yield factors and independent blocks have shown to be sufficient by Müller [24].

A simplified explanation would be to state that when blocks are fully correlated their “average number of fault” parameters are simply added together as if they were part of the same block.

$$Y_{block1\&2} = Y(\lambda_{block1} + \lambda_{block2}) \quad (38)$$

In contrast, when they are uncorrelated, their yield factors are multiplied as in (37) above. In this report the blocks are considered uncorrelated and independent.

On a side note, when combining blocks using the Poisson model (8), correlation makes no difference since the defects themselves are considered independently distributed as can be seen below:

$$Y_{block1\&2} = Y(\lambda_{block1} + \lambda_{block2}) = e^{-(\lambda_{block1} + \lambda_{block2})} = e^{-\lambda_{block1}} e^{-\lambda_{block2}} = Y(\lambda_{block1}) Y(\lambda_{block2}) \quad (39)$$

For the Negative Binomial random variable this is not the case because of the clustering.

$$Y(\lambda_{block1} + \lambda_{block2}) = \frac{1}{\left(1 + \frac{(\lambda_{block1} + \lambda_{block2})}{\alpha}\right)^{\alpha}} \quad (40)$$

$$Y(\lambda_{block1})Y(\lambda_{block2}) = \frac{1}{\left(1 + \frac{\lambda_{block1}}{\alpha}\right)^{\alpha}} \frac{1}{\left(1 + \frac{\lambda_{block2}}{\alpha}\right)^{\alpha}} \quad (41)$$

The first expression (40) is sometimes referred to as a large-area Negative Binomial distribution and the second (41) small-area Negative Binomial distribution. In an article Koren, Koren and Stapper developed a more general expression for every case in between [25].

This section has shown the theories needed to form a yield expression that requires everything in the chip to work. The statistical distribution Poisson and the Negative Binomial model have been introduced briefly with their common and most defining parameters, the average number of faults, which is very important but also very difficult to derive in practice. The block expressions have shown how to express the yield of different blocks when all blocks are required. The next section will show how this basic expression can change with added redundancy.

5 Yield enhancement techniques

5.1 Redundancy

Improving yield by introducing redundant copies was proposed by Murphy [8] only a short while after the theoretic benefits of producing more than one circuit elements per integrated chip had been established.

The general idea behind the concept of added redundancy is to lower the acceptance level of the final product tests. Stated in another way, the redundancy increases the number of acceptable outcomes that represents the product fulfilling its specified requirements.

To illustrate the basic concept, take for example a chip design that requires N identical circuit elements, at an arbitrary abstraction level, to function. Let the probability of manufacturing one circuit element successfully be

$$Y = Pr(\text{circuit element } X \text{ is fault free})$$

With no redundancy added, the probability of the entire chip being manufactured correctly can be expressed as the only acceptable outcome, when all of the circuit elements are manufactured correctly.

$$\begin{aligned} Y &= Pr(\text{chip is fault free}) = \\ &= Pr(\text{ N circuit elements are fault free}) \\ &= Pr(\text{circuit element } X_1 \text{ fault free } AND \text{ circuit element } X_2 \text{ fault free } AND \ldots \\ &\qquad \ldots AND \text{ circuit element } X_N \text{ fault free}) \end{aligned}$$

Assuming that each of the probabilities that a circuit element is being manufactured are statistically independent from the rest, this simplifies into

$$\begin{aligned} Y &= Pr(\text{chip fault free}) = \\ &= Pr(\text{circuit element } X_1 \text{ fault free}) \, Pr(\text{circuit element } X_2 \text{ fault free}) \ldots Pr(\text{circuit element } X_N \text{ fault free}) \\ &= Pr(\text{circuit element } X_i \text{ fault free})^N \end{aligned}$$

The last step follows if the circuit elements are identical from the models point of view.

A problem occurs when the number of components grows. Each generation gives room for additional logic blocks. In the case of multi core architectures, of course some of the additional blocks might be more cores, since it requires little to no extra development time to add another copy of an already designed block.

Figure 8 shows the yield of an imaginary system of only one type of blocks; each block is represented by a constant yield factor. The plot shows how the system yield changes when the number of blocks increases. Figure 8 contains ten different cases each with a block yield from left to right of 90 to 99 %.

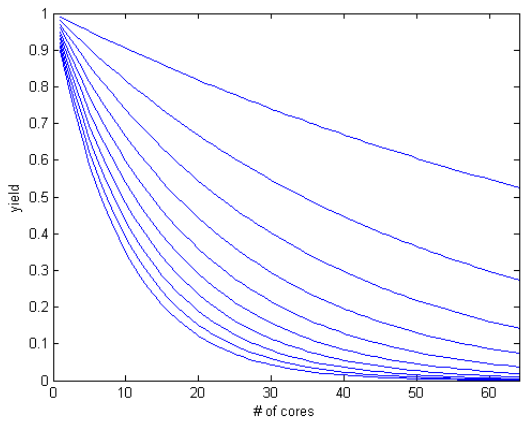


Figure 8: The combined yield of a system with multiple cores requiring all the cores to work

As can be seen, the total yield rapidly falls towards zero even for the case when 99% yielding blocks are used.

Considering instead that R redundant copies are added to the design for a total of $N+R$ number of identical circuit elements. Let X denote the number of circuit elements manufactured incorrectly. The probability that at least N copies are manufactured correctly with redundancy becomes

$$Y = Pr(\text{chip fault free}) = Pr(X \leq R) = Pr(X = 0) \; OR \; \Pr(X = 1) \ldots OR \; Pr(X = R)$$

When calculating each of the probabilities, it is necessary to consider the number of possible combinations of each of the acceptable outcomes.

The problem starts to resemble what is known as the Binomial random variable [9]. The definition of the binomial probability mass function (PDF) can be written as follows:

$$\text{Binom}(n,p) : \Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \text{ where } k = 0,1,2... \quad (42)$$

And the Cumulative Distribution Function (CDF) :

$$\Pr(X \leq x) = \sum_{k=0}^{x} \binom{n}{k} p^k (1-p)^{n-k} \quad (43)$$

To modify the binomial PDF to correspond to the example above, let

$n = N+R$

$p = 1 - \Pr(\text{circuit element } x_i \text{ fault free})$

$$Y = \sum_{k=0}^{R} \binom{N+R}{k} (1 - \Pr(\text{ciruit element } x_i \text{ is fault free}))^k \Pr(\text{ciruit element } x_i \text{ is fault free})^{N+R-k} \quad (44)$$

The circuit elements or blocks need to be identical and exchangeable with the redundant copies. In this project, the interesting circuit elements are the identical cores in a multi-core DSP cluster, and the following is an example of how the yield expression would look for such a multi-core DSP cluster.

$$Y_{\text{multi core}} = \sum_{k=0}^{R} \binom{N+R}{k} (1 - Y_{DSP})^k Y_{DSP}^{(N+R-k)} \quad (45)$$

The multi core yield factor will enter the equation representing the yield of the whole chip just like the other yield factors as follows

$$Y_{chip} = Y_{multi\ core} Y_{MEM} Y_{func1} Y_{func2} Y_{func3} = Y_0 \left(\sum_{k=0}^{R} \binom{N+R}{k} (1 - Y_{DSP})^k Y_{DSP}^{(N+R-k)} \right) \left(\prod_{\forall i} Y_{block\ i} \right)$$

For more advanced expressions that take correlations between blocks into consideration, see references [22] or [25].

5.1.1 Redundancy in memories

Redundancy is commonly used in memory circuits such as in large SRAM and DRAM arrays [23, 26]. Memory arrays are good examples of matrices of identical circuit elements.

In the 1980s, companies such as IBM sold partially good memory chips [27], that is, products containing faulty elements with functionality redefined by treating part of the chip as redundant. These memory chips contained a certain number of non-redundant memory cells. If the post-fabrication tests showed that the chip contained a fault of some sort, it was not discarded. Instead, the chip went through additional configuration to lock out the faulty chip regions, and thus a 128 k bit memory could be sold as a 64 k or 32 k bit memory. Even though the partially good products might not sell for as much as a fully functioning product, they still contributed to the return of investment for the product, which it would not have if it would have been discarded.

A similar approach is taken with processor cores of the same architecture and design that are sold rated at less than maximum capacity, for example with different nominal clock frequencies.

However, even though some circuit elements on a chip are replicated with several identical copies, and redundancy might be an effective approach to increase the potential yield, one must not forget that all components share some infrastructure such as the clock and power distribution networks. If faults during manufacturing occur in these areas, no amount of redundancy will save the chip from being discarded and adding to the yield loss statistics. Those types of faults are usually referred to as “chip kill” faults.

5.2 Wire spreading and dummy filling

Two redundancy techniques have been saved for last: wire spreading and dummy filling.

5.2.1 Wire spreading

As the name implies wire spreading aims to spread conductive wires further apart. The idea is to try to reduce their sensitivity to defects which would cause short circuits. Care needs to be taken because when wires are spread apart they may need to make additional turns or corners that are more difficult to manufacture than a straight line. Therefore, even if the critical area is reduced and the associated defect limited yield increases, the parametric yield might be affected by the new hard-to-manufacture layout structures; this effect might not be taken into account by the algorithm spreading the wires.

There is also a similar approach to reduce the open critical area by instead widening the wires to make them less sensitive to smaller defects. When the wires are widened, the available area is exactly the same as before which means that the spacing between wires decrease, in other words an inverted wire spreading occurs indirectly which increases the critical area of shorts [1,20].

Both wire spreading and wire widening are usually applied after the routing stage. The routing stage determines the shape of wires in a way that has been optimized to reach a timing goal. Afterwards, the wire spreading and widening will change the shape of these wires to optimize yield, which at the same time can not avoid having a possible negative effect on timing. The timing of the whole system must therefore be rechecked and perhaps invalidated and rerouted.

Because of the uncertain net gain of using wire spreading and widening, these methods should be used with care so as to not introduce unnecessary repetitions of the design phase.

5.2.2 Dummy filling

Dummy filling is a method used to even out the chemical-mechanical polishing (CMP) step in the manufacturing process. The polishing step is meant to even out the all the bumps and achieve a uniform thickness across the entire wafer.

The problem is that the process is not completely uniform; deviations are mainly caused by the varying metal density across the chip. When depositing material, for example the copper constituting the different metal layers, the whole wafer is processed at once. This causes metal to be deposited in both the etched metal lines and also everywhere else, and leads to a jagged uneven metal surface, which the following CMP step is meant to even out. The areas on the chip devoid of metal lines have a close to even area already and will risk ending up over polished, which affects the processing steps for the layers above [20]. The phenomenon is called dishing, and to avoid it, metal areas with no function, so called dummy fills, are inserted in hope of reducing the chances of over-polishing.

6 The proposed yield modeling approach

In this section, a method to create a yield model for a system-on-chip is proposed. One of the main requirements was to be able to predict the yield as early in the design phase as possible. The steps described in the following section have been chosen with this requirement in mind, so the abstraction level is kept at system level as much as possible. Similar approaches have been described in the past by Stapper [6], Koren [22], and Khare [29], then primarily with memories in mind.

6.1 Step I – breaking down a system into blocks

The yield enhancement method of choice for this project is redundancy. A block that is a candidate for having spare redundant copies available to replace them is considered a repairable block.

The first natural step is to separate the system design into repairable and non-repairable blocks. An example of a repairable block could be a small memory block, or anything that can be replicated with reasonable implementation penalties.

$$Y = \prod_{\forall i} Y_{non\ rep\ \mathrm{i}} \prod_{\forall j} Y_{rep\ \mathrm{j}} \qquad (46)$$

What is reasonable or not is of course something that is not set in stone, but differs from system to system. In a chip with only a single core, the whole core might be considered irreplaceable or non-repairable since duplicating the core would essentially mean an area twice the size of the original design. For a SoC with a multi-core cluster, which is the focus of this report, the individual cores are considered repairable.

Example 4.
Assume a system with a DSP multi-core cluster and five miscellaneous functional blocks f0, f1, f2, f3 and f4. The multi core cluster is considered repairable, but the functional blocks are unique and so must be considered in separate yield terms. The functional blocks could be considered repairable though, if as in the below illustration the f4 block could be a memory containing redundant sub-blocks.

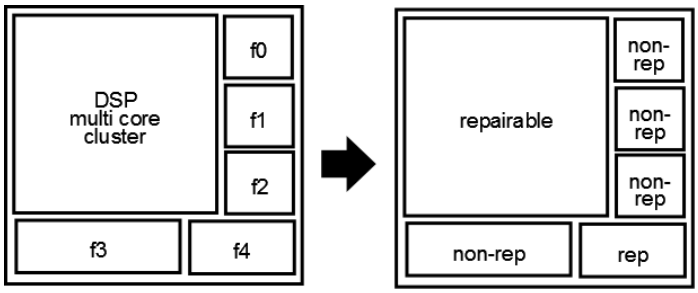


Figure 9: An imaginary system with a multi core cluster, divided into repairable and non-repairable blocks on the right

If the DSP cores themselves contain regular components, these can also be marked as repairable; prime examples are the memory areas such as instruction and data cache.

The method is repeatable for every distinguishable level of abstraction. The best example is memory blocks that can be divided into quadrants that can consist of several arrays consisting of sub-blocks further consisting of groups of words. Each abstraction level could have its own possible redundancy solution, redundant quadrants, redundant arrays, redundant sub-blocks or redundant rows and columns as described by Hampson [26].

Example 5.
Assume the same system as in example 4. The multi core cluster DSP units have an imaginary architecture like the illustration below with an instruction cache, data cache, addition/subtraction unit, instruction decode, shifting unit, some registers, a flow control unit and a multiplier/division. The DSP would then have the following yield expression

$$Y_{DSP} = Y_{I-cache} Y_{D-cache} Y_{\substack{add \\ sub}} Y_{I-decode} Y_{Shift} Y_{\substack{flow \\ ctrl}} Y_{reg} Y_{\substack{mul \\ div}}$$

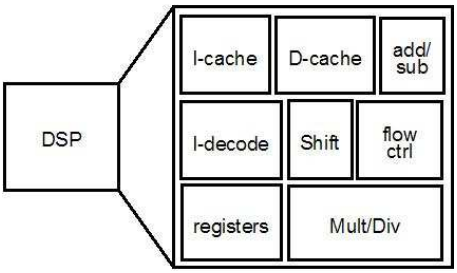


Figure 10: imaginary DSP architecture logic floor plan

The redundancy needs extra configuration circuitry that redirects the non-faulty blocks to the correct outputs which effectively masks the defective blocks. The extra configuration circuitry adds additional critical area regions, and more specifically non-repairable critical area, that need to be taken into consideration in the yield model.

Next, use the binomial random variable to form the resulting repairable yield factors. Assume no correlation between different blocks; the resulting expression should be of the form

$$Y_{rep} = Y_{config} \sum_{k=0}^{R} \binom{N+R}{k} Y_{\substack{sub- \\ block}}^{R+N-k} (1 - Y_{\substack{sub- \\ block}})^k \qquad (47)$$

N represents the required number of units that are required to work in the system specification, or in other words the number of units that would be in the non-redundant system. The variable *R* representing the number of redundant units.

Example 6.
Looking back at the system from example 4 again shown in Figure 9. The yield expression for the total System on Chip with the multi core cluster and the five functional units

$$Y_{SoC} = Y_{\substack{multi \\ core \\ cluster}} Y_{f4} \prod_{i=0}^{3} Y_{fi}$$

$$Y_{\substack{multi \\ core \\ cluster}} = Y_{\substack{DSP \\ config}} \sum_{k=0}^{R_{DSP}} \binom{N_{DSP}+R_{DSP}}{k} Y_{DSP}^{R_{DSP}+N_{DSP}-k} (1 - Y_{DSP})^k$$

$$Y_{f4} = Y_{\substack{f4 \\ config}} \sum_{k=0}^{R_{f4}} \binom{N_{f4}+R_{f4}}{k} Y_{\substack{f4 \\ sub \\ block}}^{R_{f4}+N_{f4}-k} (1 - Y_{\substack{f4 \\ sub \\ block}})^k$$

There is no real limit to in how much detail the model can describe a single block. Even though yield could be derived from a single layout structure and its corresponding critical area, it is probably more practical to choose a higher level of abstraction whenever possible, especially since the layout is more or less considered an unknown part of the yield expression.

6.2 Step II – scaling the blocks with content

In section 4.2, the theories of block yield were described for defects. The yield expression was described by a statistical distribution, either Poisson (9) or Negative Binomial (12). The Negative Binomial distribution gives a somewhat more optimistic yield estimate, assuming that defects cluster together and therefore affect fewer chips on the wafer [22].

This section uses the Negative Binomial model as it is the more general one. The parameters of the statistical distribution were the clustering parameter α and the average number of faults λ.

The clustering parameter must be received from the manufacturer and there is no real short cut or simple way to estimate it.

Deriving the average number of faults is achieved by critical area analysis by developing geometric expressions or simulations on the final layout [15]. Performing a critical area analysis for a complete system design is quite time consuming. Also, the final layout is not available when making architectural choices about adding redundant blocks to the floor plan.

However recognizing that a chip design consists of multiple identical copies of standard cells, each with the same or similar critical area, making an estimate of the critical area scaled by the number of gates in the design should be possible and intuitively a very close guess.

Therefore, the approach chosen in this report is to describe functional blocks in terms of the average number of faults for each distinguishable logic block scaled by *n* the number of standard cells the block consists of.

$$\lambda_{block} = \sum_{\forall i} n_{\substack{std \\ cell\,\mathrm{i}}} \cdot \lambda'^{std}_{cell\,\mathrm{i}} \qquad (48)$$

Example 7.

$$\lambda_{block} = n_{\substack{\mathrm{rnd} \\ \mathrm{logic}}} \cdot \lambda'^{\mathrm{rnd}}_{\mathrm{logic}} + n_{SRAM} \cdot \lambda'_{SRAM} + n_{ROM} \cdot \lambda'_{ROM} + n_{\substack{flip- \\ flops}} \cdot \lambda'^{flip-}_{flops} + \ldots$$

The largest source of error in this critical area estimate is not really accounting for the exact routing between the gates, which is most likely not equivalent in all designs unless the structure of the layout is really regular like a memory cell matrix. The intermediate routing part has to be accounted for in the “per standard cell” average number of faults most probably by an average of previous designs.

The type of yield model described in this section is meant to be used long before the final layout has even started — maybe even before the standard cell library have been developed, so the “per standard” cell average number of fault to be used to describe a block needs the ability to be rescaled based on a previous technology node’s data. An example of such a scaling was shown in [7]:

$$\lambda_{new} = \frac{A_{new}}{A_{old}} \left(\frac{w_{old}}{w_{new}} \right)^2 K \lambda_{old} \qquad (49)$$

A representing the area, *w* the minimum design feature and *K* a complexity factor which mysteriously is not mentioned in great detail, either because it is considered proprietary or because it is simply another free variable used for data fitting.

To support the approach described in this section, it must be assumed that the design will be thoroughly analyzed at the end of the design cycle, when the number of changes affecting the final layout have been reduced to almost nothing, otherwise the time consuming critical area extraction has to be repeated for every little change.

The average number of faults obtained from the critical-area analysis of the different layers is then used to derive the per-gate average number of faults by minimizing the mean square error of the difference between the two fault-rate equations as shown below:

$$\lambda_{\substack{block\\crit\\area}} - \lambda_{\substack{block\\std\\gate}} = \sum_{\forall j}\left(\int_{\forall x} A_{crit}(x)D(x)\right) - \sum_{\forall i} n_{\substack{std\\cell\,\mathrm{i}}} \cdot \lambda'^{std}_{cell\,\mathrm{i}} \quad (50)$$

$$\min\left\{E[(\lambda_{\substack{block\\crit\\area}} - \lambda_{\substack{block\\std\\gate}})^2]\right\} \quad (51)$$

Where *j* in the left summation of (50) is the index of every layer from diffusion to the top metal layer, and *i* in the right summation of (50) is the index of the standard cell type.

Note that the gate count in (50) should be that of the same design that is the target of the critical area analysis.

This circuit content scaling approach was first showed by Stapper in [7, 15] and reported in 2007 to still be in use by IBM [29]. IBM's method to capture yield apparently does not distinguish between different types of logical gates such as NAND, OR, and inverters. The reason may be that certain types of gates are used for all logic in that particular example, such as multi-level NAND gate nets. Another reason could be that the differences in critical area for the different standard cells are small enough to have insignificant effect on the total chip, consisting of billions of standard gates.

The approach described in this section assumes that λ values for the different types of standard cells are readily available from the silicon foundry. The reasons for choosing this abstraction level is again that the layout is not available when the yield model is to be used, and even if the full layout were available, then the details needed to get a yield estimate from it might reveal a bit too much information about the fabrication process through the defect density at each step. The manufacturing process is what the foundry makes its money from so the foundry might be unwilling to reveal any details regarding its implementation. The average number of faults is still a quality metric, but it is a product of both the fabrication process and the layout for the standard cell library used, making it harder to decipher any details.

The idea is to hopefully obtain a single parameter with the influence of many process steps lumped together, describing an individual standard cell from the active diffusion layer up to and including the intermediate routing layers.

6.3 Step III – global routing

The global routing which interconnects two blocks across the chip is treated differently from the standard cells in the blocks. It is difficult to associate the global interconnects with a particular standard cell, since most likely the top metal layers are reserved for the global routing [20].

Articles on critical area extraction usually describe the process with simple examples of a single straight wire, or multiple straight parallel wires of equal length [11].

In the recent technology nodes of 45 nm, each metal layer has a dedicated direction of routing, either horizontal or vertical, with little to no opportunities for notches or routing in the other directions. With the single routing direction coupled with the number of conductors which are known very early in the design process through interface specifications, the global

interconnects are simply multiple parallel wires with similar length. If the routing length in each layer can be estimated, then the critical area for the global interconnects can be almost exactly determined by the expression for *N* parallel conductors, as described previously in section 4.2.3 and repeated below:

$$A_{open}(x) = \begin{cases} L(x+(N-2)w+(N-1)s) & \text{, for } \ 0 \le x \le w \\ NL(x-w) & \text{, for } \ w \le x \le 2w+s \\ 0 & \text{, otherwise} \end{cases} \qquad (52)$$

In expression (52) the *w* and *s* represents the conductive wire width and the space between the wires respectively.

Something to keep in mind when estimating critical area is that it cannot be larger than the total area of the circuit. Extensions exist for the above expressions which take the chip width and height into consideration, by looking at the wire displacement from the middle of the circuit area. When the critical area expands until it reaches the edge of the chip, it stops expanding in this direction and continues to expand only in the opposite direction. The exact displacement from the centre of the chip for every wire might not be readily available, so instead taking the minimum of the critical area estimate and the total area will give a decent result.

$$\min\{A_{cr}, A_{tot}\} \qquad (53)$$

The critical area is only part of the equation; the full detail of the defect size distribution is then required together with the average defect density for each layer.

$$\lambda = \int_0^\infty A_{cr}(x) D(x)\, dx \qquad (54)$$

Stapper has shown [11] that the above expression can be evaluated for *N* parallel conductors and simplified into the following expression

$$\lambda = \frac{\mathrm{L}((\mathrm{N}+1)\mathrm{w}+\mathrm{Ns})\overline{\mathrm{Dx}}_0^2}{2\mathrm{w}(2\mathrm{w}+\mathrm{s})} = NL \cdot \frac{(w+s)\overline{\mathrm{Dx}}_0^2}{2\mathrm{w}(2\mathrm{w}+\mathrm{s})} + \frac{\overline{\mathrm{Dx}}_0^2 w}{2\mathrm{w}(2\mathrm{w}+\mathrm{s})} \qquad (55)$$

The expression (55) contains a variable for the average defect density of the associated layer, which might be out of reach from the manufacturer, but there are estimates on defect density provided in ITRS 2007 [30].

Even though all of these expressions are old and simplified examples, because of the strict constraints in routing in the <45 nm era, these simple expressions should be enough to fully describe most of the higher metal layers that are commonly devoted to the global interconnects. The interfaces between blocks should have been determined relatively early and a preliminary floor plan should be enough to approximate its effect on layout [31].

As an example, consider a SoC with (*N*+*R*) DSP cores and *M* miscellaneous functional blocks. All blocks, both functional blocks and DSP cores, share a larger memory block on chip for performance reasons, and each of them has an individual memory interface. If the memory interface consists of an *A*-bit-wide buss for addressing and a *D*-bit-wide bus for data transfer during a read or write operation, then the number of wires that contributes to the critical area is (*N*+*R*+*M*)(*A*+*D*) bits. An average wire length could be estimated from visual inspection of the floor plan, or a length distribution function for each layer, as the derived by de Gyvez et al [32]. Let metal layer 8, 9 and 10 have length distribution functions L_8, L_9 and L_{10}. The general idea would be for the model to be able to predict the critical area from these basic parameters.

6.4 Step IV – putting it all together

A decision needs to be made whether or not to associate the global routing yield with its corresponding DSP block. The situation might be that the global interconnects are routed on top of the neighboring DSP blocks to reach the intended destination for most cases.

The DSP block yield based on standard cells is treated as being statistically independent because it makes the yield expressions much simpler and no test data is available to prove otherwise. The simplification seems somewhat justified since the blocks are spatially separated. The global routing on the other hand spans several blocks which was just assumed to be independent. Indeed, it is difficult to intuitively justify that a few of thousands of parallel data paths are independent when they are separated only by name and a few nano meters.

Alternative 1- associate the global routing with its corresponding block

$$Y_{DSPi} = Y_{\text{std cell part i}} Y_{\text{global routing i}} \quad (56)$$

$$Y_{chip} = Y_{\text{DSP cluster}} Y_{\text{mem block}} Y_{\text{func block n}} Y_{\text{func block n+1}} \quad (57)$$

Alternative 2 - bunch all global routing into a separate yield term

$$Y_{chip} = Y_{\text{DSP cluster}} Y_{\text{mem block}} Y_{\text{func block n}} Y_{\text{func block n+1}} Y_{\text{global routing tot}} \quad (58)$$

$$Y_{\text{global routing tot}} = \prod_{\forall k} Y_{\text{global routing k}} \quad (59)$$

The positive aspect of using the first alternative is that it makes sure that the global routing is included in the redundancy calculation; and powering down a whole faulty block such as a DSP core should also affects its interconnects, including the global interconnect. The more parts considered repairable in the model, the better the estimated yield will be. The second alternative could be used if the global routing is shared, for example for a time slotted bus used for communication between blocks.

The yield expression, either (57) or (58), for the chip should now be combined with the number of chips per wafer as in (2) from section 2. For this task, the numerical algorithm described briefly in the section 3 was chosen, based on an implementation described in the original article [4]. The algorithm tries to fit as many chips as possible based on the chip width and height and the effective wafer diameter, while still obeying a few simple rules about the allowed chip placements on the wafer.

$$\text{Number of chips per wafer} = \textit{numerical algorithm}(w, h, d) \quad (60)$$

$$\text{Number of good chips per wafer} = \text{Number of chips per wafer} \cdot Y_{chip} \quad (61)$$

Now from the above equation if the cost per wafer is known, the average cost per chip can be easily calculated.

$$\text{cost per chip} = \frac{\text{cost per wafer}}{\text{Number of good chips per wafer}} \quad (62)$$

To add an area cost for any redundant part, the area per redundant block A_{red} multiplied with the number of redundant blocks R is added to the total area by breaking it down into a resulting total chip width and height increase that will keep the aspect ratio of the original chip design without any redundancy.

$$w_{new} = \sqrt{w_{old}^2 + \frac{A_{red} \cdot R}{\text{aspect ratio}}} \qquad (63)$$

$$h_{new} = w_{new} \cdot \text{aspect ratio}$$

Both the number of chips per wafer and the manufacturing yield of the chip can now be evaluated for different amounts of redundancy.

7 Application of the model on an example system and result analysis

In this section, the yield modeling approach described in sections 3 to 6 will be applied to an example SoC design containing a multi core DSP cluster. Before diving head first into the results, a few assumptions about the example SoC are listed along with motivations.

7.1 An example system

The focus of this project was to examine the profitability of using redundant cores as a yield enhancement strategy. In order not to divert attention away from the multi-core clusters effect on the number of good chips per wafer, the rest of the chip's sub-blocks will be assumed non-repairable and appear as a constant yield factor together with a gross yield.

$$Y_{SoC} = Y_0 Y_{\substack{multi\\core\\cluster}} \prod_{\forall i} Y_{fi} = Y_{\substack{multi\\core\\cluster}} \cdot \text{constant} \qquad (64)$$

This assumption does not mean that the rest of the chip's content yield is trivial to the end result. The non-repairable chip content will put a cap on how much the yield can increase with added redundancy. The cap is a fundamental limitation to probabilities which are values between zero and unity, and the product probabilities can therefore not become larger than the smallest factor.

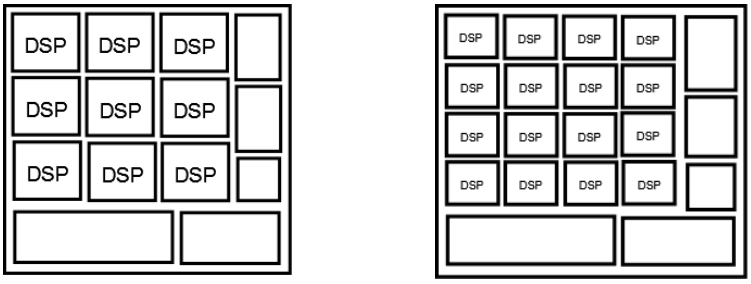


Figure 11: The example SoC topology for 9 and 16 cores. The area for the multi core cluster remains the same for both cases before redundant cores are added.

The DSP cluster will be examined at different sizes. The number of required cores N_{DSP} = {1, 2, 4, 8, 16, 32, 64} corresponds to the case without any redundant cores included. The DSP cluster area without any redundancy is assumed to be equivalent for all examples.

$$A_{tot} = A_{\substack{multi\\core\\cluster}} + A_{misc} \qquad (65)$$

$$A_{\substack{multi\\core\\cluster}} = A_{SoC} \cdot 80\% \Rightarrow A_{DSP} = A_{\substack{multi\\core\\cluster}} / N_{DSP} \qquad (66)$$

The assumption is made to more easily compare the different cases; but it does mean that the area penalty will be different from case to case. Hopefully this assumption is not overly unrealistic: it could correspond to the downscaling from one process node to the next, with enhanced performance through additional DSP cores that take up the same area as the previous generation.

The content of each DSP is assumed to be generic and the only visible part to the model is the standard cell count of each type.

$$\textit{DSP}\ \text{content}: \begin{cases} \text{number of}: \\ \text{random logic standard cells}: \mathrm{n}_{\substack{\text{rnd}\\\text{logic}}} \\ \text{memory standard cells}: \mathrm{n}_{\text{mem}} \end{cases}$$

Two different categories of standard cell are assumed: one for memory cells and one for random logic. One variant would represent the types of standard cells such as NAND, NOR-gates etc. for logic and the other variant would represent Flip-Flops, SRAM or ROM cells for memory.

The standard cell types could of course be divided up in any number of categories. It depends on the level of detail provided by the early gate count estimations. For blocks reused from previous designs, the exact gate-count of every type of inverter, NAND, half-adder might be known, but for blocks not yet implemented, a rough estimate of the total gate count and the approximate ratio of memory might be all that is available.

$$\text{new block content}: \begin{cases} \text{total gate count}: \mathrm{n}_{\text{tot}} \\ \mathrm{n}_{\substack{\text{rnd}\\\text{logic}}} = (1-\beta)\cdot \mathrm{n}_{\text{tot}} \\ \mathrm{n}_{\text{mem}} = \beta \cdot \mathrm{n}_{\text{tot}} \end{cases}$$

Regardless how the values are obtained, they will be used together with the average number of fault per standard cell variables.

$$\lambda_{\substack{\text{std}\\\text{cell}}} = \lambda'_{\substack{\text{rnd}\\\text{logic}}}\, \mathrm{n}_{\substack{\text{rnd}\\\text{logic}}} + \lambda'_{\text{mem}} \mathrm{n}_{\text{mem}} \qquad (67)$$

Example 8.

$$\lambda_{\substack{\text{std}\\\text{cell}}} = \left(1.5\cdot 10^{-7}\right)\cdot\left(10^{6}\right) + \left(2.1\cdot 10^{-7}\right)\cdot\left(10^{6}\right)$$

These scalable “average number of fault” variable values are assumed to be readily available from the manufacturer.

No actual design with a gate count can be shown in this report, but assuming a specific value might have shown results that would only have been true for that specific value. Instead ten values on the average number of faults that represent a DSP core is used in the a range below

$$\lambda_{\text{DSP}} = [0.01\,;\,0.1] \text{ average number of faults per chip}$$

7.2 Result and analysis

When putting the assumptions about the model and the example system from section 7.1 all together, the model will estimate the number of good chips per wafer. The illustrations below were created by implementing a simple version of the yield-modeling approach described in section 6 in Matlab. The results will be shown for the example system with different numbers of cores in the multi core cluster. In the following parts of this section, the example system will be referred to as "*N* core system" with *N* equal to the number of cores required according to the system specification.

The first two illustrations in Figure 12 show the single or 1 core system on the left and the 2 cores system on the right. The results for the 1 core system are somewhat expected because when blocks of mainly random logic are unique instances the yield benefit of added redundancy is overshadowed by excessive overhead of increasing to the area for the multi core to twice the size. Moreover, this simple example does not even take the extra configuration logic into account. Similarly for the system with 2 cores required to work the overhead is less but the trend undeniable with the constant reduction in chips per wafer the more redundancy added. The best amount of redundancy in these two configurations is none at all.

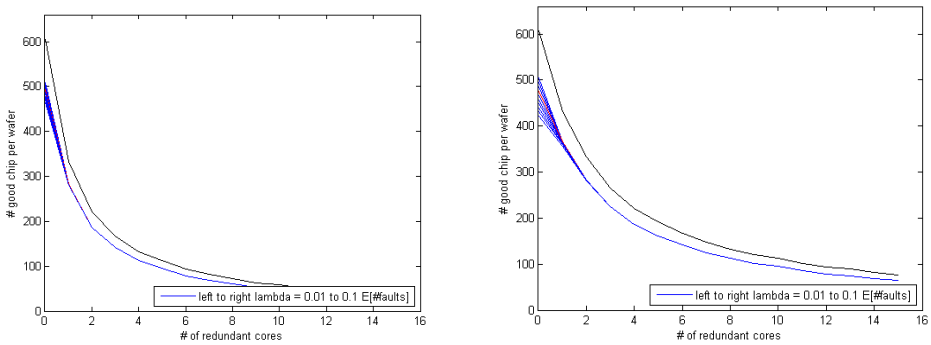


Figure 12: The number of good chips per wafer with different amount of redundant cores for example system configuration for 1 and 2 required cores.

The next two illustrations in Figure 13 are the 4 core system on the left and the 8 cores system on the right. Just as in Figure 12, the area penalty is quite large and the maximum number of chips per wafer is being reduced quite rapidly when more redundant cores are added to the system. However a few things are worth noting. The yield for the original configurations without any redundancy cores are both above 50% for all but the worst cases of the 8 cores system.

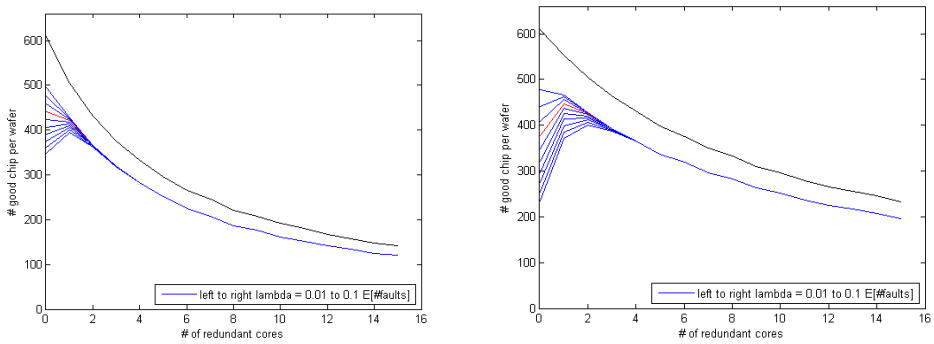


Figure 13: The number of good chips per wafer with different amount of redundant cores for example system configuration for 4 and 8 required cores.

The 4 core system is the first system configuration which shows any hint of benefit from adding redundant cores, but not for every case, and adding redundant cores might be a risky business investment considering how steeply the curve slopes downwards for adding too much redundancy.

The 8 cores system shows improvement on the number of good chips per wafer for all but one case. The 8 cores system is a case similar to the IBM Cell processor that originally had 8 cores but when used in the Playstation 3 only had 7 cores visible to the software [34].

The next two system configurations shown in Figure 14 are the 16 core system to the left and 32 core system to the right. These plots are quite similar to each other, but there are a few changes from the previous plots in Figure 12 and Figure 13.

The two most noteworthy differences are that the starting number of good chips per wafer without redundancy is now really low, which on the other hand is not too surprising since each new core is an additional source of possible error, and for every plot shown in this section the number of cores are doubled. The second thing to note is that the right side of the peak value for both systems in Figure 14 the downward slope does not look as risky as being on the left side. In other words, choosing to not have any redundancy or even adding too few redundant cores appears to be a much greater risk than having too much redundancy.

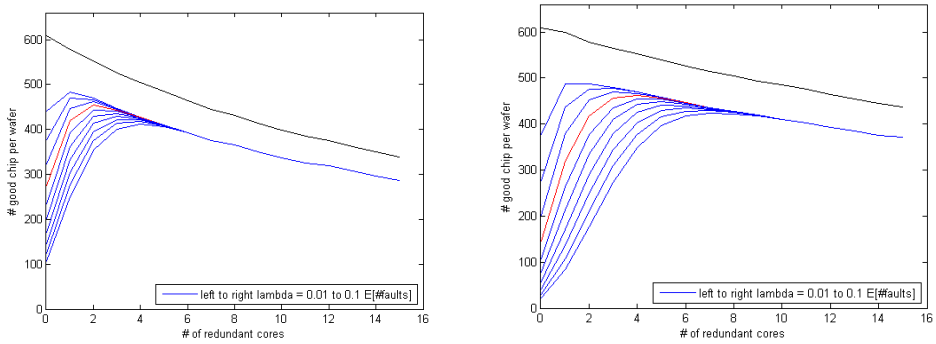


Figure 14: The number of good chips per wafer with different amount of redundant cores for example system configuration for 16 and 32 required cores.

The last example shown is for a 64 core system in Figure 15. Having redundant cores in the system seems almost mandatory at this point. The number of good chips per wafer is increased in the best case shown from a little less than 300 chips to 500 chips with only 2 redundant cores added to the system. As noted above there is a greater risk of having too little redundancy.

Multi core clusters beyond 64 cores continue in the same fashion as shown in these few examples. The general trend seems to be that for systems with at least 8 cores in their original configuration, redundant cores should definitely be considered. For larger number of cores, the question is not whether redundancy should be used or not, it is instead a question of how many redundant cores to use.

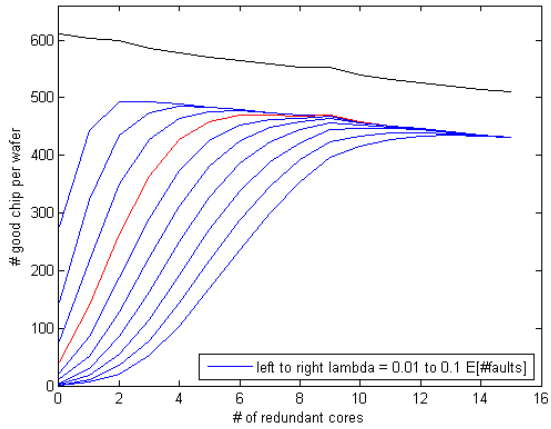


Figure 15: The number of good chips per wafer with different amount of redundant cores for example system configuration for 64 required cores.

Do remember that in this section, only a small interval of values for the average number of faults is shown; but these values have been chosen so that the even in the worst case the yield for a single core should be above 90%.

It is also worth reiterating that many of the assumptions in this section were made to give clear examples of the effects of introducing redundant cores to a multi core system, but should be carefully considered when modeling a real system.

The red line, or the fourth graph from the top, in the plots shown in Figure 12 to Figure 15 above corresponds to a value mentioned in a report by Weber [38] that some manufacturers use as a signal to stop improving their manufacturing process when the average number of faults reaches this value for their test chips. Any further optimizations were simply not cost efficient, and thus, the red line could be seen as the best case actually achievable.

The last paragraph hints that the average number of faults variable and thereby also manufacturing yield is actually a function of time. The next section 8 will briefly consider the time aspect of yield, and how redundancy will affect it.

8 Discussion and related topics

This section describes a few topics closely related to the yield modelling approach described in section 6. It discusses the added redundancy's effect in time, and how the model could be updated to capture this behaviour. At the end of the section, there is a brief discussion of the limitations of the model and some aspects that could be improved upon in future work.

8.1 Yield learning/ramp up

Yield learning is when the manufacturing yield is increased because of effort on the foundry side. For example, for defect limited yield, the defect density is decreased.

When the yield learning curve is increased during the initial months of close to zero yield, it is commonly referred to as the yield ramp phase of a manufacturing process. The aim is to have the yield ramp as early in time as possible to allow the mass production to start as soon as possible.

The importance of getting an early yield ramp originates in the field of economics, regarding a project investment's early budget which has probably set a preliminary price on the chip, or on the product of which it is a part.

Discussion in this section is based on Baker [35].

The core issue is that there is a difference between selling the product and receiving the income right at this moment, and receiving same amount at any time thereafter. The difference is the cost of lost opportunity of the income received later. If the income had been received as of this moment it could have been reinvested to increase its value compared to the unchanged value if the income was delayed.

> **Example 9.**
> Assume the yield ramp for a hypothetical product occurred 1 year after schedule but the products were sold and 100 units of currency worth of income were received.
>
> Comparing the above scenario to if the yield ramp would have happened as planned and sold for the exact same amount of 100 units of currency. At that time the income could have been invested in a bank account with an annual interest rate of for example 1%, which would have increased its value to 101 units by the time the late yield ramp would have given only 100 units.

The effect of reduction in income value is usually a key part when discussing the net present value concept (NPV). The net present value is the sum of all incomes, both positive and negative (expenses) over time for a particular project.

$$\text{Net Income} = \sum_{\forall i} I_i(t) \tag{68}$$

Every project has a starting point in time t = 0, and all incomes for t > 0 are affected by a so called discount rate, which is the same as the interest rate in the example above, only this time in the denominator.

$$NPV = \sum_{\forall i} \frac{I_i}{(1 + \text{discount rate})^a} \tag{69}$$

Here *a* represents the frequency with which the discount rate is evaluated; for example, for an annual discount rate, *a* represents the number of years from the start of the project.

NPV indicates the project relative worth. When the NPV becomes zero, the project has paid off as much as if the invested money had been put into a bank account with interest equal to

the discount for the duration of the project time. To maximize profit, the project should be terminated when the NPV is at its highest value.

If the variables in the NPV equation stay fixed and the discount rate is changed ,the discount rate at which the NPV is zero is called a project's Internal Rate of Return (IRR). The IRR seems to represent an arbitrary projects irregular income stream transformed in to an equivalent bank account investment. This transformation is done to be able to compare the profitability of similar projects.

The bank account example is a no effort investment, and is used as a base example. Another possible alternative is to dig a hole and hide your investment until you need it, this would represent an IRR = 0 and the projects NPV would most likely never gain a positive value. Thus, a project with an IRR less than that of the interest rates offered by banks and similar safe investments are considered more or less a waste of time. For the basic case, the higher valued IRR the better, although there are exceptions [35].

The important thing to remember when looking at the NPV equation is that delaying an income term in time increases the value of the a-parameter and therefore decreases the resulting magnitude. When manufacturing chips, the later the yield ramp the later the products can be sold and lower the positive income's effect towards the NPV.

Also before the yield starts to increase to levels when you can start mass-producing your integrated circuit, as a company you have no project-related products to sell and therefore have no income while still keeping the manufacturing facility and machinery running. The net income will stay negative until there is something to sell. If the yield stays low for long enough, the products sold will have been reduced so much in value over time that the net project worth will be relatively negative.

Other reasons to strive for as early a yield ramp as possible include the presence of a competing company. If there is a company *A* whose product provides similar service to a product of another company *B*, and there is a supply shortage on the market for this particular service, or the demanded quantity of products is higher than the quantity supplied, then company that launches its corresponding product first will be able to set the prices, since there is simply no other alternative available.

8.2 Modeling yield learning

The yield learning that results in a ramp up effect refers to the process engineers working to detect and suppress defect sources. Defects are found by manufacturing test structures.

Common test structures are memory arrays, which are used because of their high fault visibility. Knowing the content of the memory, each separate memory cell can be addressed individually and checked for error.

Suppressing the defect sources results in a lower defect density, and as seen in previous sections a lower defect density results in a yield increase.

In the early stages of a new technology process, there should be multiple possible sources of defects for a given fault. As the number of possible defect sources decreases, it should be easier to determine which sources are the most critical ones and spend the learning time on the right thing.

The only yield ramp modeling approach found is one described by Müller [24]:

$$D(x,(n+1)\cdot\Delta t) = (1-\text{learning rate per } \Delta t)\cdot D(x,n\cdot\Delta t)$$
$$D(x,(n+1)\cdot\Delta t) = (1-\text{learning rate per } \Delta t)^{n}\cdot D(x,0) \qquad (70)$$

Updating the timeless defect density discussed in previous sections with these simple functions will affect the average number of faults variable both the one for the global routing and the ones that are scaled with the standard cell gate counts.

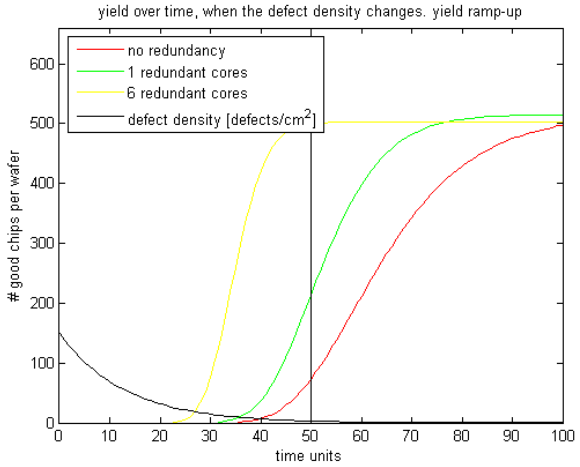


Figure 16: yield ramp for a 64 core DSP cluster, 3 configurations of redundancy. Black line defect density over time

Also for the simple example system shown in the results section will be affected by this change. Repeated here is the 64 core system from section 7 shown for a select few redundant configurations.

The black line represents not chips per wafer, but the defect density and how it changes over time. The left vertical line shows where the defect density stops improving as mentioned in previously at the end of section 7.2.

9 Future work

The main goal of this project is to develop a yield model which can help to determine the appropriate amount of redundancy. The problem could be seen as an optimization problem. But what performance metric is the correct one to optimize?

A problem with adding redundancy in some form to the current functional yield model is that the yield increases the more redundancy you have, although with diminishing returns. According to the model, the cost increase caused by the larger cluster and chip size is compensated by the higher number of sellable chips.

However, the total cost of the redundancy is more difficult to establish. In this project, the focus has been on optimizing the number of good chips per wafer, but there are other issues to consider.

How much testing time will be added to detect and configure, for every single chip manufactured, the cores which are functional and should be used? This extra processing step will increase the manufacturing cost, but it will probably be a non-reoccurring cost regardless of the number of redundant cores in the system. Hampton [26], reported a time increase in testing per good chip, but there were no real information to what factors led up to the additional time or exactly how it was determined.

How much extra research and development time will be needed to implement the reconfiguration logic and verifying that that all possible architectural configurations fulfill the original requirement specification?

How can the non-functional units or extra functioning units be prevented from interfering with the rest of the systems? How can the use of redundant cores be made invisible to the software?

What performance degradations will there be on a multi core cluster with R redundant cores? The physical topology will differ from chip to chip, although they represent the same product. This will lead to varying system performances: for example in the network on chip, a token ring network has to deal with the delay from R extra nodes, compared to a system with no redundancy. A multi hop network will have different node to node delays depending on which cores are actually used [36].

How much extra delay or power consumption will the configuration logic and the redundant copies add? Can the defective or extra unit blocks be shut off completely without affecting the rest of the system and consume little to no additional power consumption?

The above questions are difficult to answer in such a short project as this one and therefore must be considered to be outside its scope. However it is important to mention the existence of additional cost so as to not be misled by the reduction in manufacturing cost per wafer alone.

10 Conclusions/summary

The purpose of the project was, as defined in section 1, to examine the concept of redundancy for a multi core DSP cluster, in order to evaluate the profitability of adding redundant cores.

As explained in section 2, both manufacturing yield and the area of a chip directly affects the manufacturing cost of the product, which was chosen as the quality metric.

The cost function for chip area was studied in section 3 to capture the negative influence of added redundancy. The cost function estimates the maximum number of chips that can fit on a wafer for a particular chip size. When redundant cores are added, the chip area will increase, which will results in fewer chips fitting on the wafer.

To measure the positive effects of redundancy, a yield model was required. The modeling approach chosen was described in sections 4 to 6. The model focuses on defect limited yield and divides the yield into two main factors. One yield factor is based on the number of standard cells in a block. The other yield factor represents the global interconnects which are more difficult to associate with a single block.

Together, the yield model and the area cost function were used to compute the number of good chips per wafer. The fewer number of good chips per wafer, the higher the cost per chip will be.

In section 7, the model was applied to an imaginary system example with a multi core cluster of varying size. The individual DSP core blocks used in the multi core cluster had a yield above 90%. Even though the values used were not based on real product designs, the results show a clear trend.

The results in section 7.2 showed that systems with a requirement for 8 or more functional cores could benefit from adding redundant cores. The highest number of good chips per wafer was achieved when having between 10% and 20% additional redundant cores. For systems with 16, 32 and 64 cores in the basic configuration with no redundancy, there was a greater negative impact for choosing too few redundant cores than choosing too many.

In section 8 it was shown that the redundancy, while making the circuit more resilient to defects, also pushes the yield ramp back in time. The yield ramp determines the start of the mass volume production phase and so redundancy can potentially decrease the product time to market.

There are, however, still some unexplored areas that need to be researched, as seen in section 9. The nonrecurring cost and details of implementing the redundancy has not been taken into account. Future work could also focus on extending the yield model to account for parametric variation if higher abstraction level models are developed.

To use the model to produce accurate estimates would require close collaboration with a manufacturing company who would be willing to provide the required input parameters.

All in all, the concept of redundancy seems to fit perfectly in a multi core system. For a system with cluster sizes larger than 8 cores, adding redundant cores would almost have to be considered mandatory to achieve reasonable manufacturing cost.

References

In order of occurrence:

[1] Dragone, N., Guardiani, C., Strojwas A. J. (2006) Design for manufacturability in the nanometer era. In *EDA for IC Implementation, Circuit Design, and Process Technology*, Lavagno, L., Scheffer L., Martin G. (editors), pp 19-1 – 19-27. Taylor Francis Group, LCC

[2] MOSIS (2010) http://www.mosis.com

[3] Melzer, H. (2006) Smaller is Better? Maximization of Good Chips per Wafer by Co-Optimization of Yield and Chip Area, *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*

[4] de Vries, D. K. (2004) Investigation of Gross Die per Wafer Formulas, *IEEE Transactions on Semiconductor Manufacturing, vol. 18. No. 1.*

[5] Pessah, J., Digital Test Methods: Iddq tutorial, Test Technologist Team, inc www.ttt.com/downloads/iddq.pdf *(2010-05)*

[6] Stapper, C. H. (1983) Integrated Circuit Yield Statistics, *Proceedings of the IEEE, vol. 71, no. 4.*

[7] Stapper, C. H. , Patrick, J. A. , Rosner, R. J. (1993) Yield Model for ASIC and Processor Chips, *International Workshop on Defect and Fault Tolerance in VLSI Systems*

[8] Murphy, B. T. (1964) Cost Size Optima of Monolithic Integrated Circuits, *Proceedings of the IEEE, vol. 52, pp. 1537-1545*

[9] Miller, S. L. Childers, D. G. (2004) Probability and Random Processes With application to signal Processing and Communications. Elsevier Academic Press, 84 Theobald's Road, London WCTX 8RR, UK

[10] Maly, W. , Deszczka, J. (1983) Yield Estimation Model for VLSI Artwork Evaluation, *Electronics Letters, Vol. 19, No. 6*

[11] Stapper, C. H. (1984) Modeling of Defects in Integrated Circuit Photolithographic Patterns. *IBM Journal of Research and Development, vol. 28, no. 4*

[12] Webb, C. (2008) 45 nm Design for Manufacturing, *Intel Technology Journal, Vol. 12, Issue 02.*

[13] Santarini, M. (2007) A look at 45-nm-IC-design challenges, *ElectronicsWeekly.com http://www.electronicsweekly.com/Articles/2007/10/01/42290/a-look-at-45-nm-ic-design-challenges.htm*

[14] Nelson, J. E. , Zanon, T. , Brown, J. G. , Poku, O, Blanton R. D. S. , Maly, W. (2006) Extracting Defect Density and Size Distributions from Product ICs, *IEEE Design & Test of Computers.*

[15] Stapper, C. H. , Rosner, R. J. (1995) Integrated Circuit Yield Management and Yield Analysis: Development and Implementation. *IEEE Transactions on Semiconductor Manufacturing, Vol. 8, No. 2.*

[16] Statistical Process Control, www.siliconfareast.com/spc (2010-04)

[17] Moore, G. (1965) Cramming more components onto integrated circuits. *Electron, Vol. 38. pp 114-117*

[18] Sapatnekar, S. S. (2006) Static Timing Analysis. In *EDA for IC Implementation, Circuit Design, and Process Technology*, Lavagno, L., Scheffer L., Martin G. (editors), pp 6-1 – 19-16. Taylor Francis Group, LCC

[19] Kong, J-T. K. , (2004) CAD for Nanometer Silicon Design Challenges and Success, *IEEE Transactions on VLSI Systems, Vol. 12, no. 11.*

[20] Chiang, C. C. , Kawa, J. (2007) Design for Manufacturability and Yield for Nano-Scale CMOS, *Springer, P.O Box 17, 3300 AA Dordrecht*

[21] Agarwal, K. , Rao, R. , Sylvester, D. , Brown, R. (2007) Parametric Yield Analysis and Optimization in Leakage Dominated Technologies, *IEEE Transactions on VLSI Systems, Vol. 15, no. 6*

[22] Koren, I. , Koren, Z. (1998) Defect Tolerance in VLSI Circuits: Techniques and Yield Analysis, *Proceedings of the IEEE, Vol. 86, no 9*

[23] Ramadan, N. H. (1997) Redundancy Yield Model for SRAMS, *Intel Technology Journal Vol 1, Issue 2*

[24] Müller-L, G. E. (2009) A General Yield Model from Design to Product Engineering, *IEEE Transactions on Semiconductor Manufacturing. Vol. 22, no 4*

[25] Koren, I. , Koren, Z. , Stapper, C. H.(1993) A Unified Negative-Binomial Distribution for Yield Analysis of Defect-Tolerant Circuits, *IEEE Transactions on Computers, Vol. 42, No. 6*

[26] Hampson, C. W. (1997) Redundancy and High-Volume Manufacturing Methods, *Intel Technology Journal, Vol. 1, Issue 2*

[27] Stapper, C. H. , McLaren, A. N. , Dreckmann, M. (1980) Yield Model for Productivity Optimization of VLSI Memory Chips with Redundancy and partially Good Product, *IBM Journal of Research and Development, Vol. 24, No. 3*

[28] Khare, J. Feltham, D. B. I. , Maly, W. (1993) Accurate Estimation of Defect-Related Yield Loss in Reconfigurable VLSI Circuits, *IEEE Journal of Solid-State Circuits, Vol. 28, No. 2*

[29] Barnett, T. S , Bickford, J , Weger, A. J (2007) Product Yield Prediction and Critical Area Database, *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*

[30] Yield Enhancement (2007) *The International Technology Roadmap for Semiconductors, http://www.itrs.net*

[31] Koren, I. Koren, Z. (2000) Incorporating Yield Enhancement into the Floorplanning Process, *IEEE Transactions on Computers, Vol. 49, no. 6*

[32] de Gyvez, J. P. , Christie, P. (2001) Pre-Layout Prediction of Interconnect Manufacturability, *Proceedings of the 2001 international workshop on System-level interconnect prediction*

[33] Kumar, R. (2007) The Business of Scaling, *IEEE SSCS Newsletter*

[34] Sperling, E. (2006) Executive Insight: Turn Down the Heat…Please, *Electronics Design, Strategy, News, http://www.edn.com/article/465456-Turn_Down_the_Heat_Please.php*

[35] Baker, S. L. (2007) Economics Interactive Tutorials, *http://hadm.sph.sc.edu/Courses/Econ/tutorials.html*

[36] Zhang, L. Han, Y. , Xu, Q. , Li, X. W. , Li, H. (2009) On Topology Reconfiguration for Defect-Tolerant NoC-Based Homogeneous Manycore Systems, *IEEE Transactions on VLSI Systems, Vol. 17, No. 9*

[37] Milton, J. S. , Arnold, J. C. (2003) Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Science, 4th Edition, *McGraw-Hill Higher Education*

[38] Weber, C. (2004) Yield Learning and the Sources of Profitability in Semiconductor Manufacturing and Process Development, *IEEE Transactions on Semiconductor Manufacturing, Vol 17, No 4*