A New Paradigm for Trading Off Yield, Area and Performance to Enhance Performance per Wafer

Yue Gao
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, USA
yuegao@usc.edu, mb@ceng.usc.edu, yanzhiwa@usc.edu

Abstract—In this paper we outline a novel way to 1) predict the revenue associated with a wafer, 2) maximize the projected revenue through unconventional yield enhancement techniques, and 3) produce dice from the same mask that may have different performances and selling prices. Unlike speed binning, such heterogeneity is intentional by design.

To achieve these goals we overturn the traditional concepts of redundancy, and present a novel design flow for yield enhancement called "Reduced Redundancy Insertion", where spares can potentially have less area and performance than their fathers. We develop a model for the revenue associated with the new design methodology that integrates system configuration and leverages yield, area and performance. The primary metric used in this model is termed "Expected Performance per Area", which is a measure that can be reliably estimated for different system architectures, and can be maximized by using algorithms proposed in this paper. We present theoretical models and case studies that characterize our designs, and experimental results that validate our prediction. We show that using Reduced Redundancy can improve wafer revenue by 10-30%.

I. INTRODUCTION

A. General Discussion of Pertinent Concepts

For chip manufacturers, maximizing profit per silicon wafer is of key concern. Profit per wafer is dictated by yield, area and performance of the design. In this paper we describe a novel redundancy insertion technique to enhance the value of these metrics at the architectural level. First, we informally describe our viewpoint of these metrics.

1. Yield (Y): Normally yield is defined as the fraction of “good” or “marketable” dice sampled from a large quantity of manufactured dice. One can achieve a high yield by using techniques such as ECC [1], TMR [2], and a mature CMOS technology. These techniques often have a negative effect on other metrics of concern, such as dice area. In our work we view yield as the average fraction of “good” or “marketable” dice identified from a wafer. Thus, we are focused on yield/area (Y/A).

2. Area (A): The area of the design determines the number of dice that can fit on a single wafer. Also, a design with a larger area is more likely to achieve higher performance.

3. Performance (P): Performance can be measured in many ways, depending of the functionality of the system, and influences the selling price of the chip. For a microprocessor, performance is often measured in MIPS or clock frequency; for an audio device it might be measured in terms of sampling rate, bit-accuracy or MOS score [3]. In our case, performance is measured in terms of the appropriate metrics for the application.

In this paper, our focus is to describe a technique termed Reduced Redundancy Insertion (RRI) and optimization methods during the design stage to achieve near maximal values of projected profit per silicon wafer. We begin with a brief review of some of the previous related work.

B. Conventional Redundancy Insertion

While there are many ways to enhance yield and yield/area, we will not review those that are unrelated to our work. Our approach is loosely based on an extension of the work of Mirza-Aghatabar et al. [4, 5, 6]. These papers justify the utilization of redundancy and give a detailed review of previous work in this field. The approach taken by Mirza-Aghatabar et al. is to take a block of sequential logic, and 1) partition it into sub-modules, 2) replicate sub-module M_i N_i times (N_i ≥ 0), where the N_i replicated modules are considered to be spares or redundant modules (we refer to the collection of M_i and its spares as stage i), 3) insert steering logic that consists of switches, forks and joins between stages, and 4) activate one “good” module in each stage, identified by the use of scan-paths, BIST and software tools including ATPG, then disable the other modules at that stage. The objective function is to either maximize yield (Y) and/or yield per area (Y/A). Fig. 1 depicts the concepts just discussed, where after testing, activated modules are indicated in dark solid outlines and deactivated modules are indicated in gray dashed outlines.

Despite achieving maximum Y/A, this redundancy scheme has several drawbacks, which are addressed in this paper:
1) For immature technologies, modules with large area that cannot be easily partitioned have very low yield. Their spares naturally also have very low yield. To achieve acceptable overall Y/A, multiple spares are required, but this adds complexity to the steering logic, which reduces performance. Test time also increases, since more modules need to be tested.
2) After fabrication, as technology matures and yield improves beyond a certain threshold, the spares become wasted silicon space. To eliminate such waste, new masks must be produced, which is a very expensive process.
C. Reduced Redundancy Insertion

A Reduced Redundant Module (RRM) \( M'_i \) is a spare module for \( M_i \) that has less area, less performance, but not less functionality than \( M_i \). The process of inserting RRM is referred to as RRI. Reduced Redundancy (RR) capitalizes on the fact that modern systems have large silicon areas dedicated to enhancing performance [7], and this trend is predicted to remain even in the multi-core era [8]. Hence, for modules in these systems, performance enhancement features can potentially be scaled back or abandoned to form RRM. This paper will analyze the associated tradeoffs.

One can make a valid claim that for some designs using a module with degraded performance will jeopardize the operation of the entire system. For many situations, however, this is not the case. Below we list several ways that a system can operate properly with little or no reconfiguration when some modules are replaced with their RR counterparts.

- **Pipelines**
  In a conventional pipeline, performance degradation of any stage can be coped with by lowering the clock frequency.

- **Asynchronous Logic/Handshaking Interfaces**
  In asynchronous circuits modules communicate with each other through handshake protocols, and thus each module can take an arbitrary amount of time to finish computation. RRI can be easily applied in such environments.

- **Commercial Microprocessors**
  - **The Instruction Scheduler Module**
    The instruction scheduler inside modern microprocessors includes single-cycle wake up, wide scheduling width and scheduling heuristics to boost performance [9]. A simpler scheduler can be used without reconfiguration due to the handshake policies with the execution units.
  - **Execution Units**
    Execution units, such as ALUs, can be designed in different ways, balancing area and performance. The integer execution unit in Pentium 4 [10, 11] employs carry-merge trees, domino logic and operates at the 2x frequency domain. These features can be removed.
  - **The Entire Core**
    At the coarsest granularity, each core can have its own spare [12]. A RR core will have smaller area and degraded performance. Such a system would resemble ARM's big.LITTLE architecture [13], where a A15 core is coupled with a lightweight A7.

- **Intrinsic Error-Tolerant Systems**
  Some systems can inherently tolerate errors at the application level, such as video/audio decoding. Shin et al. [14] exploits these characteristics and explicitly reduces area in the design stage while sacrificing computation accuracy. These modules are perfect candidates for RRI.

In some cases, in order to use RRM reconfiguration is needed. We integrate this reconfiguration overhead into the area of the RRM. Details regarding synchronization of the RRM with the entire system in terms of protocols and/or clock distribution are beyond the scope of this paper.

D. Reduced Redundancy Insertion Design Flow

RR calls for a revision in the conventional yield improvement design flow (Fig. 2a). In the RRI design flow (Fig. 2b) the design is passed onto RR analysis, where the algorithms presented in this paper are executed, returning more ideal RR configurations. This information is then fed back to the circuit designers, who will attempt to redesign modules while satisfying the RR analysis requirements.

![Image of Figure 2](image-url)

**Figure 2. Conventional and Reduced Redundancy Design Flow**

RR analysis and RR redesign can go through several iterations if the circuit designer cannot satisfy the requirements of the RR analyzer. After reaching some point of agreement, the design will be manufactured, tested and configured so that failed modules are replaced by their spares. Although some chips are sold as lowered performance versions, which stems from RRI and perhaps also speed binning, the total revenue per wafer achieved with RRI can often significantly outperform the traditional redundancy insertion schemes.

II. PROBLEM DEFINITION

A. Case Study - Prelude

We examine a simplified case study to serve as an introduction to RRI. Consider an abstract microprocessor that consists of a cache module \( M_0 \) and a core module \( M_1 \), illustrated in Fig. 3, manufactured on a wafer of size \( A_w \). The area and yield of \( M_i \) are \( A_i \) and \( Y_i \) respectively. \( Y_i \) is estimated by yield models. By the use of spare rows/columns, ECC and recent software/OS level fault tolerant techniques [15], even in environments of high defect densities and process variations, the cache \( (M_0) \) can achieve a high yield. \( M_1 \) however, does not share the uniformity of the cache module. The average performance of \( M_1 \) is \( P_1 \), the value of which is estimated by executing architectural or post-synthesis benchmark simulations, and will serve as the baseline performance value.

![Image of Figure 3](image-url)

**Figure 3. A 2-Module abstract microprocessor**

Over the initial years of production when the technology is immature or progressing through the tedium of yield learning, a non-negligible fraction of the manufactured \( M_1 \) will produce errors due to defects, noise and process variations. This causes the entire chip to be discarded. Thus, \( M_1 \) needs to be more robust, and one way to accomplish this is via redundancy. This can be done using coarse-grained core-level redundancy [12, 16]. We explore this solution as an introductory example. Assume only one spare core can be used for \( M_1 \). Multiple spares are prohibited due to the complexity of the steering logic, testing logic and testing time. \( M_0 \) will not have a spare since its yield \( Y_0 \) is high. Fig. 4 displays three versions of \( M_i \) design to be examined: the bare design, the conventional redundancy design and the new RRI design.
Version 1 Design (No Redundancy)

The first version has no redundancy. From this point on we assume that the yield of any module is independent of the yield of other modules, and therefore the yield of the entire chip is $Y_0 Y_1$. The $Y/A$ of the chip is $Y_0 Y_1/(A_0+ A_1)$. For each silicon wafer of size $A_w$, the number of “good” dice with performance $P_1$ is approximately $A_w Y_0 Y_1/(A_0+ A_1)$.

Version 2 Design (Traditional Redundancy, 1 Spare Only)

Version 2 will have one spare for $M_1$ which is identical to $M_1$. Therefore, a chip will only fail if $M_2$ fails, or both $M_1$ and its spare fail. The yield of the entire chip is $Y_0 [1-(1-Y_1)^2]/(A_0+2A_1)$. For each wafer of size $A_w$, the number of “good” dice with performance $P_1$ is $A_w Y_0 [1-(1-Y_1)]/(A_0+2A_1)$.

Version 3 Design (Reduced Redundancy, 1 Spare Only)

The third design will utilize RRI. The RRM, denoted by $M_r'$, has area $A_r'$, where $A_r'<A_1$, performance $P_r'$, and yield $Y_r'$. $M_r'$ has all the functionality of $M_r$, but $P_r'<P_1$ due primarily to the reduction in area. The decrease in area leads to an increase in yield, which holds for all well-known defect-oriented yield models. To summarize: $A_r'<A_1$, $P_r'<P_1$ and $Y_r'>Y_1$. The overall performance can no longer be evaluated as a constant, since it depends on which parts are functional (see Table I).

<table>
<thead>
<tr>
<th>Version 1</th>
<th>Version 2</th>
<th>Version 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1 (M1)</td>
<td>Module 1 (M1)</td>
<td>Module 1’ (M’1)</td>
</tr>
<tr>
<td>Area: $A_1$</td>
<td>Area: $A_1$</td>
<td>Area: $A_1$</td>
</tr>
<tr>
<td>Performance: $P_1$</td>
<td>Performance: $P_1$</td>
<td>Performance: $P_1$</td>
</tr>
<tr>
<td>Yield: $Y_1$</td>
<td>Yield: $Y_1$</td>
<td>Yield: $Y_1$</td>
</tr>
</tbody>
</table>

Figure 4. Three design alternatives for $M_1$.

<table>
<thead>
<tr>
<th>$M_r$</th>
<th>$M_r'$ (RRM)</th>
<th>Probability/Yield</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>Good</td>
<td>$Y_1 Y_r'$</td>
<td>$P_1$</td>
</tr>
<tr>
<td>Good</td>
<td>Bad</td>
<td>$(1-Y_1) Y_r'$</td>
<td>$P_r'$</td>
</tr>
<tr>
<td>Bad</td>
<td>Good</td>
<td>$(1-Y_1) Y_r'$</td>
<td>$P_1$</td>
</tr>
<tr>
<td>Bad</td>
<td>Bad</td>
<td>$(1-Y_1) Y_r'$</td>
<td>$P_r'$</td>
</tr>
</tbody>
</table>

Table I. Performance for different configurations of $M_r$ and $M_r'$.

For a wafer of size $A_w$, a total of $A_w (A_0+ A_1+A_r')$ dice will be manufactured. The expected number of dice/wafer that operate at the nominal performance $P_1$ is $Y_0 Y_1 A_w/(A_0+ A_1+A_r')$. Similarly, the expected number of dice that will operate at the lowered performance of $P_r'$ is $Y_0 (1-Y_1) Y_r' A_w/(A_0+ A_1+A_r')$.

The revenue each dice generates depends on various non-deterministic commercial factors. In our work we reasonably assume that revenue is linearly correlated to performance, thus to maximize revenue we should “extract” the most performance out of each single wafer. We can approximate this objective function by the expression shown in Eq. (1).

$$ A_w Y_0 \left( \frac{Y_1 P_1}{A_0+ A_1+ A_r'} + \frac{(1-Y_1) Y_r' P_r'}{A_0+ A_1+ A_r'} \right) \quad (1) $$

To help justify the use of Eq. (1), we introduce the concept of Expected Performance per Area ($E(P)/A$), which is the expected performance divided by total area. For versions 1 and 2 designs, performance is constant. Therefore the expectation of performance is constant. $E(P)/A$ in this case is simply $Y/A$ multiplied by nominal performance $P_1$. With RR, the expectation of performance is defined as:

$$ \sum \text{Yield (Chip Performance = } P_i) \times P_i $$

For the version 3 design, the value of $E(P)/A$ is:

$$ \frac{Y_0 (Y_1 P_1 + (1-Y_1) Y_r' P_r')}{A_0 + A_1 + A_r'} $$

Now Eq. (1) can be rewritten as $A_w E(P)/A$. Since $A_w$ is constant, maximizing $E(P)/A$ will maximize total revenue per wafer. For other formulations of wafer revenue, Eq. (1) would need to be adjusted accordingly.

B. Formal Formulation

Extrapolation to a system with one simplex module ($M_0$), and $n$ redundancy-applicable modules ($M_1$, $M_2$, ..., $M_n$) introduces some key complications that will be analyzed next. We assume a generic system organized as follows: $M_0$ still has size $A_0$ and yield $Y_0$, while $M_k$ ($1 \leq k \leq n$) has size $A_k$ and yield $Y_k$. $M_0$ will not have a spare. For $M_r-M_n$ we assume the following:

1. Each module can have either one or no spare. The spare can be identical to the original module as in traditional redundancy insertion or smaller in area than the original module as in RRI.
2. For each module, either the original or its spare, if it exists, must be fault free to ensure system operation.
3. All reconfiguration overhead required for utilizing the spare is incorporated in the area of the spare.
4. The sole spare for $M_k$, denoted as $M_k'$, has size $A_k'$ ($A_k' \leq A_k$) and yield $Y_k'$ ($Y_k' \geq Y_k$). If $A_k'=0$ then the spare does not exist.

The golden performance of the system $P_{\text{gold}}$ is the reference performance when $M_0-M_n$ are all functional. Later, parameters related to $M_0$ will be integrated into the equation as a common term. The probability of $P_{\text{gold}}$ being achieved is $\prod_{i=1}^{n} Y_i$.

The chip will be discarded if there is a module where both the original and its spare have failed, or the original has failed and there is no spare. The probability of stage $k$ failing is denoted as $B_k$. When $M_k$ is not protected i.e. $A_k' = 0$, then $B_k = (1-Y_k)$. When $M_k$ has one spare, i.e. $0 < A_k' \leq A_k$ then $B_k = (1-Y_k)(1-Y_k')$. According to the inclusion-exclusion theory, the probability of a chip failing due to failures in any stage is:

$$ B = \sum_{k=1}^{n} B_k - \sum_{i=1}^{n} B_i B_j + \sum_{i<j}^{n} B_i B_j B_k - \cdots + (-1)^{n+1} \prod_{i=1}^{n} B_i $$

When the chip is still functional, i.e. no stage has failed, but $P_{\text{gold}}$ cannot be achieved, the chip will operate at a certain degraded performance with the probability of $1 - \prod_{i=1}^{n} Y_i - B$.

The value of the degraded performance ($P_{\text{degrade}}$) is dependent on which spare modules are in use. Thus in reality it cannot be fixed as a constant. However, to avoid the combinatory sets of values to represent $P_{\text{degrade}}$, we take an extremely pessimistic approach, that is, of all the $P_{\text{degrade}}$ values that the system can achieve by using any combination of the spares, choose the minimum. If one or more RRM is used in place of the original, the chip is said to perform at $P_{\text{degrade}}$. This guarantees that we will not overestimate $E(P)/A$ in later calculations. This also adheres to practical marketing considerations, where only two types of chips need to be priced, those with performance $P_{\text{gold}}$ and those with the lowered performance $P_{\text{degrade}}$.

The expected performance of the entire chip including $M_0$. 
Traditional Redundancy
the circuit design, layout and manufacturing process.

In Eq. (2), when \( A' = A \), then traditional redundancy is deployed for \( M_e \), while \( 0 < A' < A \) indicates that RRM is used for \( M_e \). \( A' = 0 \) implies that no redundancy is used for \( M_e \).

The following sections of this paper will provide insight on the maximization of E(P)/A for this generic system. To show a quantitative example of the benefits of RR, we return to the example in Fig. 3. Table II summarizes the parameters of the system. The relationship between \( P' \) and \( A' \) is \( P' = \sqrt{A'} \) according to Pollack's Rule. We plot the E(P)/A for the three designs of Fig. 4 in Fig. 5.

Table II. Area, Yield and Performance parameters

<table>
<thead>
<tr>
<th>( A_0 )</th>
<th>( Y_0 )</th>
<th>( A_1 )</th>
<th>( Y_1 )</th>
<th>( P_l )</th>
<th>( A'1 )</th>
<th>( P'1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.9</td>
<td>5</td>
<td>0.5</td>
<td>( \sqrt{A_1} )</td>
<td>Adjustable</td>
<td>( \sqrt{A'} )</td>
</tr>
</tbody>
</table>

As seen in Fig. 5, the E(P)/A function for the RR design is not monotonic, and at the peak point RR exhibits about a 17% increase in E(P)/A compared to traditional redundancy.

To augment the above example, we later present Theorem 1 based on the same system depicted in Fig. 3.

C. Analysis of the E(P)/A Function
To maximize E(P)/A, we note that Eq. (2) is a function of area (\( A \)), yield (\( Y \)), and performance (\( P \)). Intuitively, as the area of a module decreases, its yield should increase and its performance decrease. We refer to the relationship between performance and area as the P-A function, which is architecture dependent. The relationship between area and yield is determined by the yield model. The conclusions drawn from our work are valid regardless of these detailed relations as long as they are well behaved. All well-known yield models can be used in our work, but for simplicity we will use the Poisson yield model:

\[
Y = e^{-d \cdot A}
\]

where \( d \) is a constant dependent on the technology and defect densities. The values of \( A \), \( Y \), and \( P_{gold} \) are predetermined by the circuit design, layout and manufacturing process.

By adjusting \( A' \), we can obtain different values of \( Y' \). If \( A' > A' \) are set, \( B \) can be calculated. \( P_{degrade} \) is also dependent on \( A' \), but this relationship requires further investigation. Assuming this relationship is known, we will only need to manipulate \( A' \) to maximize E(P)/A. Section III will lay out the construction of \( P_{degrade} \) and introduce the maximization procedure for E(P)/A.

Theorem 1: Consider the system shown in Fig. 3, and the aforementioned Poisson yield model. If performance is modeled as a single term polynomial function of area, i.e. \( P' = KA'^{R} \) where \( K > 0 \) and \( R > 0 \), RRI will enhance the E(P)/A value compared to traditional redundancy when \( A' > A' \). Theorem 1 provides some fundamental insight regarding RRI. That is, RRI is most beneficial when 1) the target module has a large area, 2) the defect density is high, and 3) the performance penalty from decreasing module area is low.

III. Maximization of E(P)/A

A. Models of Degraded Performance
The elusive relationship between \( P_{degrade} \) and \( A' \), which is defined by the function \( H \) in Eq. (3), determines the maximization methodology of E(P)/A.

\[
P_{degrade} = H(A'_1, A'_2, A'_3, ..., A'_n)
\]

In this paper we explore two \( P_{degrade} \) formulations for two types of systems: serial and parallel. We believe these two models can cover most basic architectures. For other system configurations, \( P_{degrade} \) should be either approximated by one of our two models, or derived by extending our models.

1. Pipeline/Bottleneck Performance Model
We assume that \( M_1-M_n \) are organized in a linear unidirectional pipeline. We first define a P-A function \( P_l = F(A') \) for each \( M_e \). \( F(A') \) is the performance of the entire chip when \( M_e \) is replaced by \( M_e' \). Fig. 6 illustrates the system configuration for measuring \( F(A') \), with disabled modules shaded in grey. Our algorithm assumes \( F(A') \) is given as input, where in reality the \( F(A') \) would be derived either from empirical estimations or architectural simulations. By definition, \( P_{gold} \) is achieved when \( A' = A \) \( \forall \ x \), i.e.

\[
P_{gold} = F_1(A_1) = F_2(A_2) = \cdots = F_n(A_n)
\]

2. Additive Performance Model
For the second model, we assume that \( M_1-M_n \) operate in parallel and contribute independently to the overall system performance, which is evaluated by taking the sum of all individual module performance values. This can be easily extended to a weighted sum function. Again we define a P-A function \( G(A_e) \) for each module \( x \) (\( M_e \)), which represents the performance rating of \( M_e \). By definition \( P_{gold} = \sum_{x=1}^{n} G_x(A_x) \). The worst case \( P_{degrade} \) is now defined as:

\[
P_{degrade} = \sum_{x=1}^{n} G_x(A' x) + \sum_{x=1}^{n} G_x(A_x)
\]

\[\text{such that } A'_x = 0 \]

B. Maximizing E(P)/A

1. Pipeline/Bottleneck Performance Model
Our method for maximizing E(P)/A starts by finding the optimal values of \( A', A'_2, ..., A'_n \). It consists of an outer loop
and a kernel optimization procedure. The outer loop finds the best-suited stages \( S_R \subseteq \{1, 2, ..., n\} \), where redundancy is appropriate. In other words, we have \( 0 < A' \leq A_i \) when \( i \in S_R \) and \( A' = 0 \) when \( i \notin S_R \) for each \( i \in \{1, 2, ..., n\} \). The kernel optimization solves the Simplified Redundancy Area (SRA) optimization problem, i.e. finding the optimal \( A_i' \) value for each \( i \in S_R \) when the set \( S_R \) is given.

We first describe the kernel, which aims to find the optimal \( A_i' \) values in a \( |S_R| \)-dimensional space, where \( |S_R| \) denotes the cardinality of \( S_R \). We effectively reduce the search space of \( A_i' \) values by exploiting the following theorem:

**Theorem 2:** For stages in the set \( S_R \), the optimal area values of the redundant spares satisfy:

\[
F_i( A_i' ) = f_j( A_i' ), \quad \text{for} \quad \forall i, j \in S_R.
\]

**Proof:** Let us suppose that \( \exists i, j \in S_R \) such that \( F_i( A_i' ) < F_j( A_j' ) \) in the optimal solution of the SRA optimization problem. We are going to prove that by reducing \( A_j' \) such that \( F_i( A_i' ) = F_j( A_j' ) \), the \( E(P)/A \) value calculated by (2) will increase. This will contradict the claim that \( F_i( A_i' ) < F_j( A_j' ) \) can exist in the optimal solution of the SRA optimization problem.

When reducing the \( A_j' \) value (still we have \( F_i( A_i' ) < F_j( A_j' ) \)), \( P_{\text{degrade}} < F_j( A_j' ) \) will remain unchanged. Only the \( B \) value will change in the numerator of (2) during the reduction of \( A_j' \). We know that \( B_j \) will decrease when reducing \( A_j' \). We can derive the derivative of \( B \) with respect to \( B_j \) as:

\[
\frac{\partial B}{\partial B_j} = 1 - \sum_{i \in \{1, 2, ..., n\} \setminus \{j\}} B_i + \sum_{i \in \{1, 2, ..., n\} \setminus \{j\}, j < k} B_i B_k \cdots + (-1)^{n+1} \prod_{i \in \{1, 2, ..., n\} \setminus \{j\}, j < k} B_i.
\]

Notice that \( \sum_{i \in \{1, 2, ..., n\} \setminus \{j\}} B_i = 1 \) is essentially the probability that both the original module and the spare fail in at least one stage in the set of stages \( \{1, 2, ..., n\} \setminus \{j\} \). Since the probability is less than 1, \( \partial B / \partial B_j \) is positive. In other words when we reduce the \( A_j' \) value, \( B \) will decrease, and therefore the numerator of (2) will increase. On the other hand, the denominator of (2) will decrease. Since the values of the numerator and denominator of (2) are greater than zero, it follows that the \( E(P)/A \) value from (2) will increase when \( A_j' \) decreases.

By exploiting the above observation, we successfully reduce the search space for finding the optimal \( A_i' \) values from the original \( |S_R| \)-dimensional space to a curve in which all the \( F_i( A_i' ) \) values are identical. We then utilize the ternary search algorithm, which is an extension of the well-known binary search algorithm, to find the optimal \( F_i( A_i' ) \) values, and subsequently, the optimal \( A_i' \) values. An upper bound and a lower bound are required for performing the ternary search algorithm. For each \( i \in S_R \), the lower bound of \( F_i( A_i' ) \) is 0 (also note that \( F_i(A_i') = 0 \)), while the upper bound of \( F_i( A_i' ) \) is \( F_i(A_i) \). We use \( E(P)/A |_{A_i' = 0} \) to denote the maximum \( E(P)/A \) value for a given \( S_R \), which is achieved by the kernel procedure.

The outer loop sequentially removes appropriate candidate elements in the subset \( S_R \) to determine the optimal set \( S_R^{(k)} \). In summary, the complete method for maximizing \( E(P)/A \) under the pipeline performance model is outlined in Algorithm 1.

**Algorithm 1:** Near-Optimal Redundancy Insertion and Sizing Algorithm under the Pipeline/Bottleneck Performance Model.

**2. Additive Performance Model**

The heuristic for maximizing \( E(P)/A \) under the additive performance model is rather straightforward: start with the module \( M_i \) with the largest area that already visited and set \( A_i' = A_i \). Gradually decrease \( A_i' \) until \( E(P)/A \) does not increase, or the lower bound has been reached. After the maximal point is reached, check if deleting the spare will help improve overall \( E(P)/A \), if so remove the spare by setting \( A_i' = 0 \). Mark the module as visited, and repeat until all modules are visited.

**IV. EXPERIMENTAL RESULTS**

**A. Pipeline/Bottleneck Performance Model**

For the pipeline model, we first present a 5-module system that mimics the AMD Bulldozer [17] in terms of area composition. The Bulldozer dedicates roughly 5% of the chip layout area to instruction fetch with branch prediction (\( M_1 \)), 11% to decode logic (\( M_2 \)), 15% to execution units (\( M_3 \)), 12% to memory access organization such as load/store queues (\( M_4 \)), and the remaining 57% to cache and other functions (\( M_5 \)).

System configurations for the fixed original modules (\( A_{\text{Ori}}, Y_{\text{Ori}} \)) and calculated RRM (\( A_{\text{RRM}}, Y_{\text{RRM}} \)) are listed in Table III. In this example, the overall yield of the chip (\( Y \)), which reflects technology immaturity, is \( Y_{\text{Ori}} \cdot Y_{1} \cdot Y_{2} \cdot Y_{3} \cdot Y_{4} = 0.2 \). The \( P/A \) function parameters conform to Pollack's Rule, with offsets to ensure that the original module performances are equal.

<table>
<thead>
<tr>
<th>Module</th>
<th>( A_{\text{Ori}} )</th>
<th>( Y_{\text{Ori}} )</th>
<th>( A_{\text{RRM}} )</th>
<th>( Y_{\text{RRM}} )</th>
<th>( P/A ) Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_1 )</td>
<td>0.57</td>
<td>0.83</td>
<td>0.57</td>
<td>0.19</td>
<td>( Y_{\text{P degrade}} = 3.87 )</td>
</tr>
<tr>
<td>( M_2 )</td>
<td>1.11</td>
<td>0.66</td>
<td>0.66</td>
<td>0.84</td>
<td>( Y_{\text{P degrade}} = 0.56 )</td>
</tr>
<tr>
<td>( M_3 )</td>
<td>1.45</td>
<td>0.57</td>
<td>0.57</td>
<td>0.76</td>
<td>( Y_{\text{P degrade}} = 0.65 )</td>
</tr>
<tr>
<td>( M_4 )</td>
<td>1.12</td>
<td>0.64</td>
<td>0.64</td>
<td>0.82</td>
<td>( Y_{\text{P degrade}} = 0.50 )</td>
</tr>
<tr>
<td>( M_5 )</td>
<td>5.38</td>
<td>0.38</td>
<td>0.38</td>
<td>0.82</td>
<td>( Y_{\text{P degrade}} = 0.41 )</td>
</tr>
</tbody>
</table>

\( Y_{\text{P degrade}} \) can be set to any number; it does not influence the increase percentage of \( P/A_{\text{RRM}} \).

The RRM area values (\( A_{\text{RRM}} \)) are computed using Algorithm 1. We compare the RR design to a baseline design using optimal traditional redundancy, that is, while limiting each module to have at most 1 identical spare, choose the optimal configuration using methods from [4]. The \( E(P)/A \) of the RRI design (\( E(P/A_{\text{RRI}}) \)) is 20.57% higher than that of the baseline design (\( E(P/A_{\text{Traditional}}) \)).

In Fig. 7 we plot the \( E(P)/A \) increase percentage of RRI designs over the baseline for different \( Y \) values. When \( Y \) is as low as 0.1, i.e. the defect density is high, RRI outperforms the baseline, which is already vastly superior over the original design [4], by over 50%. When technology matures and \( Y \) increases, the benefit of RRI diminishes, but RRI never underperforms the baseline design when \( Y \geq 0.8 \). Another important observation is that for \( Y \geq 0.67 \), the baseline design
will shed all redundancy, i.e. traditional redundancy is no longer helpful. However, even when traditional redundancy cannot improve Y/A, RRI still has room for 10% improvement when $Y = 0.7$ in terms of E(P)/A.

improvements over the baseline design (right on Table VI), and observe that the conservatively estimated E(P)/A value of RRI outperforms the baseline by over 10% when $Y=0.6$. Once again we see that when $Y>0.6$, the RRI can outperform the bare design when the traditional redundancy design cannot.

Figure 7. The E(P)/A values with respect to the original yield

Clearly the effectiveness of RRI also heavily depends on the $P-A$ functions. We compare systems described on the left of Table IV with different $P-A$ function parameters ($m$). The plot on the right of Table IV shows that regardless of the original yield, the benefit of RRI will start to diminish as the $P-A$ function reaches linearity, which is achieved when $m=1$.

Table IV. Effects of performance characteristics on RRI E(P)/A

Next we consider a 7-module system configuration (Table V) to demonstrate the effectiveness of the algorithm for larger problem sizes. The lower bound of $A_{RRM}$ is set to $A_{Ori}/5$.

Table V. Experimental results for a 7-module system

To detail the design space exploration process of the RRI configurations ($A_{RRM}$), Fig. 8 plots the $P-A$ functions for each module. The original designs occupy the leftmost point on each curve. Moving right along the curves corresponds to employing RRMs with smaller area. The E(P)/A of the our RRI design is 19% higher than that of the baseline.

Figure 8. Design exploration under the pipelined performance model

B. Additive Performance Model

We present a 10-module arbitrary system configuration (Left on Table VI) for the additive performance model. The lower bound of $A_{RRM}$ is set to $A_{Ori}/5$. We plot the E(P)/A

V. CONCLUSIONS

In this paper we presented a novel redundancy insertion technique termed Reduced Redundancy Insertion (RRI) that overcomes some of the drawbacks of conventional redundancy. Instead of using spares that are identical to the original modules, we appropriately choose spares that trade-off smaller area and higher yield at the cost of degraded performance.

REFERENCES