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

Yue Gao Melvin A. Breuer Yanzhi Wang
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.

This work was sponsored in part by the NSF under grant CCF-1018869

978-3-9815370-0-0/DATE13/©2013 EDAA

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 ($N_i \geq 0$), 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.

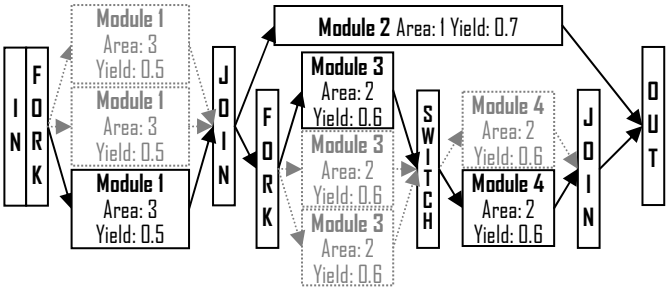


Figure 1. Redundancy used in a traditional way

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 wastage, new masks must be produced, which is a very expensive process.

C. Reduced Redundancy Insertion

A **Reduced Redundant Module (RRM)** M'_x is a spare module for M_x that has less area, less performance, but not less functionality than M_x. 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.

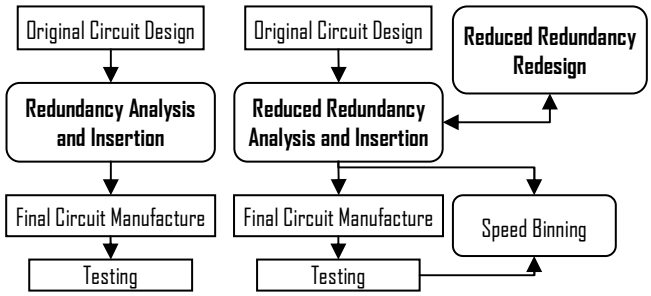
One can make a valid claim that for some designs using a module with degraded performance will jeopardize the operation of the entire system. For many situations, however, this is not the case. Below we list several ways that a system can operate properly with little or no reconfiguration when some modules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



(a) Conventional Redundancy Insertion *(b) Reduced Redundancy Insertion*

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.

Module 0 (M_0): Cache (Area A_0)	Module 1 (M_1): Core (Area A_1, Performance P_1)

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_1 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_0Y_1. The Y/A of the chip is $Y_0Y_1/(A_0+A_1)$. For each silicon wafer of size A_w, the number of "good" dice with performance P_1 is approximately $A_wY_0Y_1/(A_0+A_1)$.

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_0 fails, or both M_1 and its spare fail. The yield of the entire chip is $Y_0[1-(1-Y_1)^2]$, and the Y/A of this design 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: $A_wY_0[1-(1-Y_1)^2]/(A_0+2A_1)$.

Version 3 Design (Reduced Redundancy, 1 Spare Only)

The third design will utilize RRI. The RRM, denoted by M'_1, has area A'_1, where $A'_1<A_1$, performance P'_1 and yield Y'_1. M'_1 has all the functionality of M_1, but $P'_1<P_1$ 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: $A'_1<A_1$, $P'_1<P_1$ and $Y'_1>Y_1$. The overall performance can no longer be evaluated as a constant, since it depends on which parts are functional (see Table I).

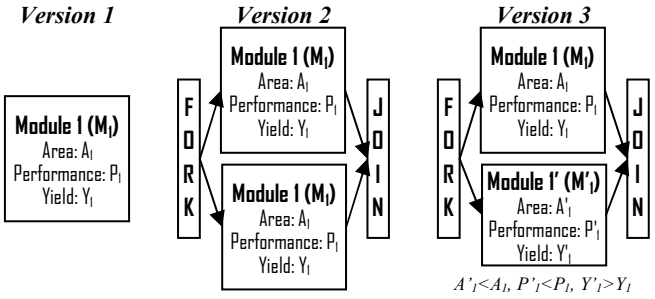


Figure 4. Three design alternatives for M_1

Table I. Performance for different configurations of M_1 and M'_1

M_1	M'_1 (RRM)	Probability/Yield	Performance
Good	Good	$Y_1\,Y'_1$	P_1
Good	Bad	$Y_1\,(1-Y'_1)$	P_1
Bad	Good	$(1-Y_1)\,Y'_1$	P'_1
Bad	Bad	$(1-Y_1)(1-Y'_1)$	0

For a wafer of size A_w, a total of $A_w/(A_0+A_1+A'_1)$ dice will be manufactured. The expected number of dice/wafer that operate at the nominal performance P_1 is: $Y_0Y_1A_w/(A_0+A_1+A'_1)$. Similarly, the expected number of dice that will operate at the lowered performance of P'_1 is $Y_0(1-Y_1)Y'_1A_w/(A_0+A_1+A'_1)$.

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).

$$A_wY_0\left(\frac{Y_1P_1}{A_0+A_1+A'_1}+\frac{(1-Y_1)Y'_1P'_1}{A_0+A_1+A'_1}\right) \quad (1)$$

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:

$$\sum_{\forall i} Yield(Chip\ Performance = P_i) \times P_i$$

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

$$\frac{Y_0[Y_1P_1+(1-Y_1)Y'_1P'_1]}{A_0+A_1+A'_1}$$

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 ($1\le k\le n$) has size A_k and yield Y_k. M_0 will not have a spare. For M_1-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 ($A'_k \le A_k$) and yield Y'_k ($Y'_k \ge Y_k$). If $A'_k = 0$ then the spare does not exist.

The golden performance of the system P_{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_{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. $0<A'_k\le A_k$ 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:

$$B = \sum_{i=1}^{n} B_i - \sum_{i<j}^{n} B_iB_j + \sum_{i<j<k}^{n} B_iB_jB_k \ldots + (-1)^{n+1}\prod_{i=1}^{n} B_i$$

When the chip is still functional, i.e. no stage has failed, but P_{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 ($P_{degrade}$) 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 $P_{degrade}$, we take an extremely *pessimistic* approach, that is, of all the $P_{degrade}$ 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 $P_{degrade}$. 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_{gold} and those with the lowered performance $P_{degrade}$.

The expected performance of the entire chip including M_0:

$$Y_0\left[\prod_{i=1}^{n} Y_i\, P_{gold} + \left(1 - \prod_{i=1}^{n} Y_i - B\right) P_{degrade}\right]$$

The E(P)/A of the entire chip is given by Eq. (2), and serves as the primary objective function in this paper.

$$\frac{Y_0\left[\prod_{i=1}^{n} Y_i\, P_{gold} + (1 - \prod_{i=1}^{n} Y_i - B) P_{degrade}\right]}{\sum_{i=0}^{n}(A_i + A'_i)} \quad (2)$$

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

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'_1 and A'_1 is $P'_1 = \text{sqrt}(A'_1)$ 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

A_0	Y_0	A_1	Y_1	P_1	$A'1$	P'_1
10	0.9	5	0.5	sqrt(A_1) = 2.23	Adjustable	sqrt(A'_1)

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.

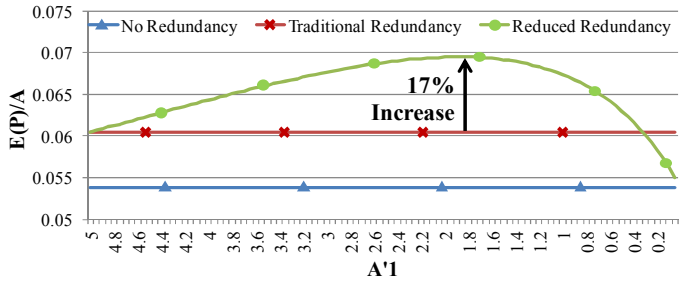


Figure 5. **E(P)/A** values with respect to the area of the spare A'_1

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, implied in the expectation) 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:

$$Yield = e^{-d \cdot Area}$$

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

By adjusting A'_1-A'_n, we can obtain different values of Y'_1-Y'_n. If A'_1-A'_n are set, B can be calculated. $P_{degrade}$ is also dependent on A'_1-A'_n, but this relationship requires further investigation. Assuming this relationship is known, we will only need to manipulate A'_1-A'_n 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. $P'_1 = KA'^{R}_1$ where $K>0$ and $R>0$, RRI will enhance the E(P)/A value compared to traditional redundancy when $A_1>A'_1>R/d$.

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'_1-A'_n, which defined by the function H in Eq. (3) determines the maximization methodology of **E(P)/A**.

$$P_{degrade} = H(A'_1, A'_2, A'_3, \ldots, A'_n) \quad (3)$$

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 uni-directional pipeline. We first define a *P-A* function $P_x = F_x(A'_x)$ for each M_x. $F_x(A'_x)$ is the performance of the **entire chip** when **only** M_x is replaced by M'_x. Fig. 6 illustrates the system configuration for measuring $F_x(A'_x)$, with disabled modules shaded in grey. Our algorithm assumes $F_x(A'_x)$ is given as input, where in reality the $F_x(A'_x)$ would be derived either from empirical estimations or architectural simulations. By definition, P_{gold} is achieved when $A'_x=A_x \ \forall x$, i.e. $P_{gold} = F_1(A_1) = F_2(A_2) = \cdots = F_n(A_n)$

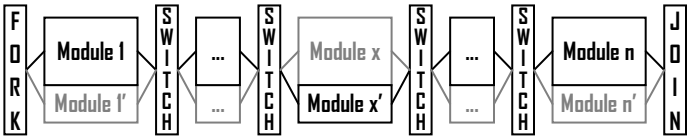


Figure 6. The pipeline hardware model for measuring $F_x(A'_x)$

The clock rate of a pipeline is bounded by its slowest stage. Thus we can reasonably assume that $P_{degrade}$ is constrained by the **RRM** that degrades performance the most, i.e.

$$P_{degrade} = min(F_x(A'_x)\ \forall 1 \le x \le n\ such\ that\ A'_x \neq 0)$$

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 $G_x(A_x)$ for each module x (M_x), which represents the performance rating of M_x alone. By definition $P_{gold} = \sum_{x=1}^{n} G_x(A_x)$. The worst case $P_{degrade}$ is now defined as:

$$P_{degrade} = \sum_{\substack{\forall 1 \le x \le n \\ such\ that\ A'_x \neq 0}} G_x(A'_x) + \sum_{\substack{\forall 1 \le x \le n \\ such\ that\ A'_x = 0}} G_x(A_x)$$

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'_1, A'_2, ..., A'_n. It consists of an *outer loop*

and a *kernel optimization procedure*. The outer loop finds the best-suited stages $\boldsymbol{S_R} \subseteq \{1, 2, ..., n\}$, where redundancy is appropriate. In other words, we have $0<A'_i \leq A_i$ when $i \in \boldsymbol{S_R}$ and $A'_i = 0$ when $i \notin \boldsymbol{S_R}$, 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 $i \in \boldsymbol{S_R}$ when the set $\boldsymbol{S_R}$ is given.

We first describe the kernel, which aims to find the optimal A'_i values in a $|\boldsymbol{S_R}|$-dimensional space, where $|\boldsymbol{S_R}|$ denotes the cardinality of $\boldsymbol{S_R}$. We effectively reduce the search space of A'_i values by exploiting the following theorem:

Theorem 2: For stages in the set $\boldsymbol{S_R}$, the optimal area values of the redundant spares satisfy:

$$F_i(A'_i) = F_j(A'_j), \qquad \text{for } \forall i, j \in \boldsymbol{S_R}.$$

Proof: Let us suppose that $\exists i, j \in \boldsymbol{S_R}$ 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 $F_i(A'_i)<F_j(A'_j)$), $P_{degrade} \leq F_i(A'_i)$ 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 calculate the derivative of B with respect to B_j, we obtain

$$\frac{\partial B}{\partial B_j} = 1 - \sum_{i \in \{1,2,...,n\}\backslash\{j\}} B_i + \sum_{i,k \in \{1,2,...,n\}\backslash\{j\}, i<k} B_i B_k \ldots + (-1)^{n+1} \prod_{i \in \{1,2,...,n\}\backslash\{j\}} B_i$$

Notice that $\sum_{i \in \{1,2,...,n\}\backslash\{j\}} B_i - \sum_{i,k \in \{1,2,...,n\}\backslash\{j\}, i<k} B_i B_k \ldots + (-1)^n \prod_{i \in \{1,2,...,n\}\backslash\{j\}} B_i$ is essentially the probability that *both* the original module and the its spare fail in *at least* one stage in the set of stages $\{1, 2, ..., n\}\backslash\{j\}$. 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 $|\boldsymbol{S_R}|$-dimensional space to a curve in which all the $F_i(A'_i)$ values are identical. Then we 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 $i \in \boldsymbol{S_R}$, 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 $\mathrm{E(P)/A}|_{\boldsymbol{S_R}}^{opt}$ to denote the maximum E(P)/A value for a given $\boldsymbol{S_R}$, which is achieved by the kernel procedure.

The outer loop sequentially removes appropriate candidate elements in the subset $\boldsymbol{S_R}$ to determine the optimal set $\boldsymbol{S_R^{(k)}}$. 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.

Initialize $\boldsymbol{S_R^{(0)}} \leftarrow \{1, 2, \ldots, n\}$.
Perform kernel optimization procedure to find the optimal A'_i value of each $i \in \boldsymbol{S_R^{(0)}}$.
For k from 0 to $n-1$:
  For each $j \in \boldsymbol{S_R^{(k)}}$:
    Perform kernel optimization procedure to find the optimal A'_i value of each $i \in \boldsymbol{S_R^{(k)}}\backslash\{j\}$.
  Find j with the maximum $\frac{\mathrm{E(P)}}{\mathrm{A}}\Big|_{\boldsymbol{S_R^{(k)}}\backslash\{j\}}^{opt}$ value.
  $\boldsymbol{S_R^{(k+1)}} \leftarrow \boldsymbol{S_R^{(k)}}\backslash\{j\}$.
  $A'_j \leftarrow 0$.
Find k with the maximum $\frac{\mathrm{E(P)}}{\mathrm{A}}\Big|_{\boldsymbol{S_R^{(k)}}}^{opt}$ value, and find the corresponding A'_i values.

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 not 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_0). System configurations for the fixed original modules (A_{Ori}, Y_{Ori}) and calculated RRM (A_{RRM}, Y_{RRM}) are listed in Table III. In this example, the overall yield of the chip (Y), which reflects technology immaturity, is $Y_0 \times Y_1 \times Y_2 \times Y_3 \times Y_4 = 0.2$. The *P-A* function parameters conform to Pollack's Rule, with offsets to ensure that the original module performances are equal.

Table III. Experimental results for a 5-module system

	A_{Ori}	Y_{Ori} (d=0.037)		A_{RRM}	Y_{RRM}	*P-A* Function	
M_0	A_0=57	Y_0=1*		-	-	-	
M_1	A_1=5	Y_1=0.83		A'_1=1.19	Y'_1=0.96	$P_1=A_1$^0.5+4.64	
M_2	A_2=11	Y_2=0.66	Y=0.2	A'_2=4.71	Y'_2=0.84	$P_2=A_2$^0.5+0.56	P_{gold} = 3.87
M_3	A_3=15	Y_3=0.57		A'_3=7.44	Y'_3=0.76	$P_3=A_3$^0.5	$P_{degrade}$ = 2.64
M_4	A_4=12	Y_4=0.64		A'_4=5.38	Y'_4=0.82	$P_4=A_3$^0.5+0.41	
$EPA_{Traditional}$ = 0.0166				EPA_{RRI} = 0.0201 (20.57% Increase)			

* Y_0 can be set to any number; it does not influence the increase percentage of EPA_{RRI}

The RRM area values (A_{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 (EPA_{RRI}) is 20.57% higher than that of the baseline design ($EPA_{Traditional}$).

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<0.8. Another important observation is that for Y>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.

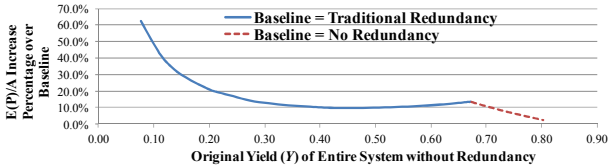


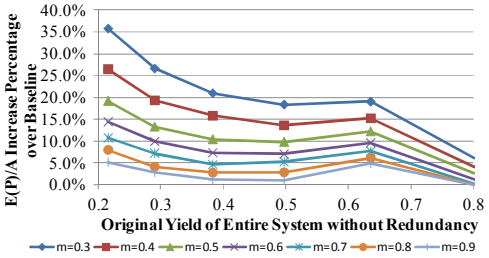
Figure 7. The E(P)/A values with respect to the original yield

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

	A_{Ori}	*P-A*
M_0	A_0=57	-
M_1	A_1=5	$P_1=A_1\text{^}m+j$
M_2	A_2=11	$P_2=A_2\text{^}m+k$
M_3	A_3=15	$P_3=A_3\text{^}m$
M_4	A_4=12	$P_4=A_3\text{^}m+n$

**m is the dominating factor in the P-A function. Pollack's Rule estimates m = 0.5 for the entire chip*



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 be A_{Ori} /5.

Table V. Experimental results for a 7-module system

	A_{Ori}	Y_{Ori}		A_{RRM}	Y_{RRM}	*P-A* Function	
M_0	A_0=15	Y_0=1		-	-	-	
M_1	A_1=6	Y_1=0.75		A'_1=4.4	Y'_1=0.81	$P_1=A_1\text{^}0.95$	
M_2	A_2=5	Y_2=0.79		A'_2=1.0	Y'_2=0.95	$P_2=A_2\text{^}0.5+3.25$	
M_3	A_3=5	Y_3=0.79	Y=0.3	A'_3=1.5	Y'_3=0.93	$P_3=A_3\text{^}0.6+2.86$	P_{gold} = 5.49
M_4	A_4=4	Y_4=0.83		A'_4=0.9	Y'_4=0.96	$P_4=A_4\text{^}0.6+3.19$	$P_{degrade}$ = 4.11
M_5	A_5=4	Y_5=0.83		A'_5=1.4	Y'_5=0.93	$P_5=A_5\text{^}0.7+2.85$	
M_6	A_6=4	Y_6=0.83		A'_6=3.6	Y'_6=0.84	$P_6=A_6\text{^}1.6-3.70$	
$EPA_{Traditional}$=0.0602				EPA_{RRI}= 0.059263 (19.28% Increase)			

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.

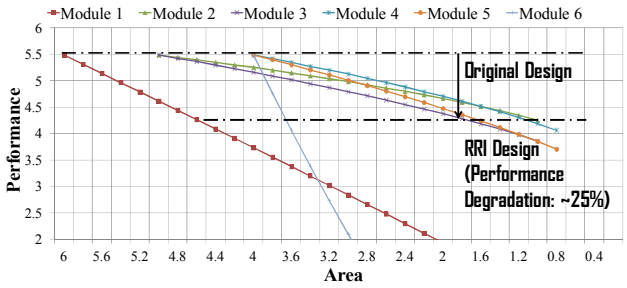


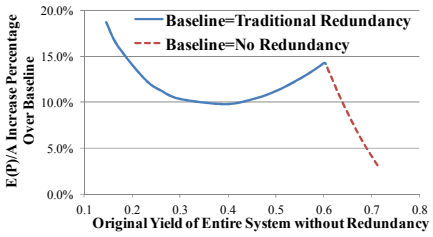
Figure 8. Design exploration under the pipelined performance model

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 $A_{Original}$/5. We plot the 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.

Table VI. Experimental results for a 10-module system

	A_{Ori}	*P-A* Func.
M_0	A_0=35	-
M_1	A_1=10	$P_1=A_1\text{^}0.6$
M_2	A_2=9	$P_2=A_2\text{^}0.5$
M_3	A_3=8	$P_3=A_3\text{^}0.55$
M_4	A_4=7	$P_4=A_4\text{^}0.4$
M_5	A_5=6	$P_5=A_5\text{^}0.6$
M_6	A_6=5	$P_6=A_6\text{^}0.6$
M_7	A_7=4	$P_7=A_7\text{^}0.8$
M_8	A_8=3	$P_8=A_8\text{^}0.7$
M_9	A_9=2	$P_9=A_9\text{^}0.9$



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*. Under such schemes, each wafer may produce dice that have different revenues, and we must co-optimize yield, area and performance to achieve maximum revenue per wafer. For this we introduce a new metric called **Expected Performance per Area**, along with algorithms to maximize this function.

In the future, we plan to integrate additional cost parameters into our E(P)/A formulations, such as industrial concerns, including testing and characterization costs.

REFERENCES

[1] J. Chang et al., "The 65nm 16mb on-die l3 cache for a dual core multi-threaded xeon processor," *Symp. on VLSI Circuits*, 2006.

[2] R. E. Lyons et al., "The use of triple modular redundancy to improve computer reliability," *IBM Journal of Research and Development*, 7(2):200–209, 1962.

[3] H. Zhu and M. A. Breuer, "A framework for the analysis of error-tolerance," *IEEE Design and Test Magazine*, 25: 168-177, 2008.

[4] M. Mirza-Aghatabar et al., "Algorithms to maximize yield and enhance yield/area of pipeline circuitry by insertion of switches and redundant modules," *DATE*, 2010.

[5] M. Mirza-Aghatabar et al., "SIRUP: switch insertion in redundant pipeline structures for yield and yield/area improvement," *ATS*, 2009.

[6] M. Mirza-Aghatabar et al., "Theory of logical partitioning of yield/area maximization using redundancy," *IEEE Workshop on DFM&Y*, 2011.

[7] M. A. Breuer, "Hardware that produces bounded rather than exact results," *Design Automation Conf.*, 2010.

[8] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," *Computer*, *IEEE*, 41(7):33–38, 2008.

[9] M. Anders et al., "A 6.5 GHz 130 nm single-ended dynamic ALU and instruction scheduler loop," *ISSCC Dig. Tech. Papers*, 410–411, 2002.

[10] D. Boggs et al., "The microarchitecture of the Intel Pentium 4 processor on 90nm technology," *Intel Technology J.*, 8(01), 2004.

[11] S. Wijerantne et al., "A 9GHz 65nm Intel Pentium 4 Processor Integer Execution Core," *IEEE Solid-State Circuits Conf.*, 2006.

[12] N. Aggarwal et al., "Configurable isolation: building high availability systems with commodity multi-core processors," *ISCA*, 2007.

[13] P. Greenhalgh, "Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7," *ARM White Paper*, 2011.

[14] D. Shin and S. Gupta, "Approximate logic synthesis for error tolerant applications," *DATE*, 2012.

[15] H. Hsuing et al., "Salvaging Chips with Caches beyond Repair," *DATE*, 2012.

[16] L. Zhang et al., "Defect tolerance in homogeneous manycore processors using core-level redundancy with unified topology," *DATE*, 2008.

[17] T. Fischer et al., "Design Solutions for the Bulldozer 32-nm SOI 2-Core Processor Module in an 8-Core CPU," *Int'l. Solid State Circuits Conf.*, 2011.