ILP-Based Identification of Redundant Logic Insertions for Opportunistic Yield Improvement During Early Process Learning

Tuck-Boon Chan[1], Wei-Ting Jonas Chan[2] and Andrew B. Kahng[2,3]
[1]Qualcomm Technologies, Inc., San Diego, CA 92121
[2]ECE and [3]CSE Departments, UC San Diego, La Jolla, CA 92093
tuckboon@qti.qualcomm.com, {wechan, abk}@ucsd.edu

***Abstract*—As semiconductor technology advances, leading-edge product companies must rapidly improve yield for designs that seek to reach mass production while early in the adoption of a new technology node; otherwise, products may be unviable in the marketplace. In this paper, we are the first to study the possible mitigation by *opportunistic, last-stage* redundant logic insertion to mitigate yield loss in early advanced-node production. We describe a yield estimation methodology, and propose an integer linear programming (ILP)-based optimization of redundant logic insertion for opportunistic yield optimization. In our approach, we first identify potential logic clusters for replication by top-down application of multilevel FM partitioning. We then select promising clusters whose replication maximizes design yield without hurting design timing. Our experimental results show the potential for significant yield improvement with minor timing impact.**

I. Introduction

Defect-limited yield is always a challenge in new technology nodes because manufacturing recipes are still being refined and processing steps are not tightly controlled. Low yield during the adoption stage of a new technology node significantly impacts product schedules. Defect-limited yield can be improved by duplicating logic gates (redundancy), but this approach has not been seriously considered in mature nodes because it incurs obvious gate area overheads. However, in advanced technology nodes, achievable placement utilization is decreasing due to increasing placement and routing congestion, constrained pin accessibility, and complex design rules [6]. As a result, redundancy insertion becomes feasible. Figure 1 qualitatively summarizes interactions among design parameters and design metrics. When we have more redundant cells, yield is potentially improved due to the redundancy. However, more cells imply that (replicated) cell clusters have more cuts – i.e., terminals – and thus consume more whitespace and timing slack while worsening congestion. Decreased whitespace, decreased timing slack, and worse congestion then limit insertion of redundant cells.

Our present paper focuses on opportunistic replication to mitigate yield loss due to random defects. (Parametric yield is beyond scope; we assume that it has been addressed by proper signoff criteria.) We propose an algorithm that exploits multiple facets of design slack – area, timing, routing congestion – that may exist in late stages of physical implementation. In our approach, we add redundant logic cells to a design and then add multiplexers (MUX cells) to switch between original logic cells and redundant logic cells. Furthermore, we consider timing and congestion as we add the redundant logic and MUX cells.

We assume that the control logic to select clusters can be realized using programmable memories. With extra testing effort, designers can reprogram the control logic to select non-defective logic. We leave the testing strategy and controller design for future work, and focus solely on the strategy to insert redundancy.

A. Previous Works

For random defect analysis, previous works use *critical area* as a surrogate to evaluate yield. Ghaida et al. [11] propose a infrastructure of critical area evaluation considering cell layouts. Papadopoulou [23] proposes a framework to calculate critical area using Voronoi diagrams. Since manufacturing yield has always been a major determinant of die cost, many previous works have investigated yield improvement from different perspectives. Nardi and Sangiovanni-Vincentelli [21] propose a synthesis framework to improve yield by reducing the number of instances of standard cells with higher critical areas. Wo et al. [24] consider block-level critical areas during architecture optimization. Iizuka et al. [15] and Bourai and Shi [2] reduce critical area by relaxing layout spacings

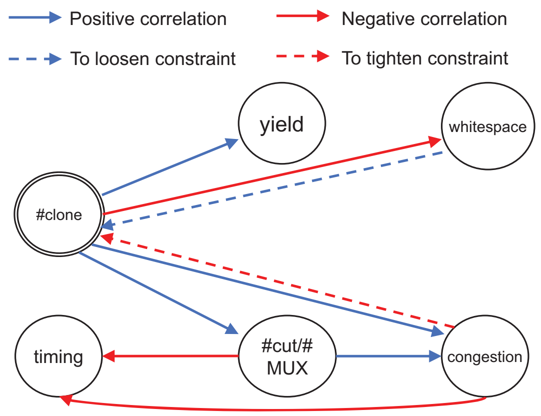

Fig. 1: The interactions among yield, amount of redundancy, and other design parameters.

of standard cells at the cost of higher area. In the arena of placement and routing (P&R), yield improvement has been achieved by a wide variety of approaches: (1) whitespace modulation during detailed placement [1]; (2) track and layer assignment with critical-area awareness during detailed routing [18] [8] [19]; and (3) post-route optimization to reduce critical area by widening wires [7], inserting redundant vias and wires [5] [20] [3], and non-tree routing [16]. Although introduction of redundant logic is considered in the literature, previous works focus on majority voting to reduce single-event upset. By contrast, here our goal is to provide *opportunistic* instance-level redundancy insertion when utilization and timing criticality allow this.[1]

B. Problem Statement and Proposed Methodology

We state our problem as follows. Our objective is to minimize random-defect yield loss. A problem instance consists of a routed design, timing constraints, and a post-redundancy insertion target utilization. We seek to determine a *post-engineering change order* (post-ECO) design that maximizes redundant logic area while meeting timing and utilization constraints.

Algorithm 1 Overview of our proposed physical implementation flow for redundancy insertion.

```
Input: Original routed design
Output: Post-ECO design with redundancy, signed-off timing and yield evaluation
1: Extract design parameters (timing, topologies, and cell masters)
2: Generate cluster candidates by iteratively calling MLPart (using RentCon)
3: Use ILP to identify clusters for replication (maximizing yield, subject to timing
   constraints)
4: Netlist editing and MUX insertion in P&R tool
5: ECO place and route
6: Post-ECO timing optimization
7: Signoff and yield evaluation
8: return Layout with redundancy, signed-off timing and evaluated yield
```

We describe our proposed methodology in Algorithm 1. We start from the routed design and extract netlist, critical paths, and available whitespace. Based on the extracted information, we

[1]*Razor* [10] is a self-repair mechanism for parametric timing error. Although it can improve parametric yield, it addresses a different yield loss mechanism and use case.

generate cluster candidates and identify optimum combination of clusters. We then edit the netlist and duplicate clusters. We add MUX cells to the output nets of clusters. The P&R tool uses the edited netlist to execute ECO placement and route. We apply timing optimization and yield evaluation after the ECO flow.

In the rest of this paper, we first describe our yield model to evaluate the benefit of inserting redundant logic in Section II. Section III then presents our heuristics to identify logic clusters for redundancy. We present experimental results in Section IV, and conclude in Section V. Table I summarizes the notations that we use in this paper.

TABLE I: Description of notations used in our formulation.

Term	Meaning
i	grid index in yield analysis
k	cluster index
j	cell index in a cluster
F_{nand}	Defect rate of a NAND2 cell
U_i	metal utilization of grid i
λ	Poisson exponent used in the yield model
Y, Y_{BEOL}, Y_{FEOL}, Y_{MUX}, Y_{redun}, $Y_{non-redun}$	design yield, BEOL yield, FEOL yield, MUX yield, and yield w/ and w/o redundancy
A_{redun}, A_{MUX} A_{chip}, A_{init}, A_k	areas of redundant logic, MUX cells, chip footprint, cells in input design and the k^{th} cluster
n, m	index of timing paths and timing points
N	number of extracted critical paths
M_n	number of timing points of n^{th} path
K	number of candidate clusters
c_k	logic cluster
C_k	binary variable of selection of a cluster
D_{MUX}	MUX cell delay
SL_n	path slack of n^{th} path
SL_{min}	minimum slack after redundancy insertion
$P_{m,n}$	binary indicator of whether m^{th} timing point of n^{th} path has a MUX
D	target utilization in post-ECO design

II. Yield Improvement by Exploiting Redundancy

We now describe our yield model and our methodology to improve yield by adding redundant logic cells.

A. Yield Impact Evaluation During Early Process Learning

For random defects, the probabilities of failure in redundant and non-redundant logic area are independent. Therefore, we can calculate design yield with redundancy as

$$\begin{aligned} Y = & Y_{non-redun} \cdot (Y_{MUX} \cdot Y_{redun}^2 + \\ & 2 \cdot Y_{MUX} \cdot Y_{redun}(1 - Y_{redun})) \\ = & Y_{non-redun} \cdot Y_{MUX} \cdot Y_{redun} \cdot (2 - Y_{redun}) \end{aligned} \quad (1)$$

where Y is design yield, $Y_{non-redun}$ is yield of non-redundant area, Y_{redun} is yield of redundant area, and Y_{MUX} is yield of MUX area (for logic redundancy). The first term in the parentheses denotes the yield while original and redundant clusters have no defects, and the second term denotes the yield while exactly one of the original and redundant clusters is defective. For a design with area A, random defect yields can be modeled by the Poisson yield model [27]

$$Y = e^{-\lambda \cdot A} \quad (2)$$

where λ is a process-dependent Poisson exponent. Given a design of area A, if we duplicate A_{redun} area of the design, with MUX area A_{MUX}, yield of the design with redundancy is given by

$$\begin{aligned} Y = & Y_{non-redun} \cdot (Y_{MUX} \cdot Y_{redun}^2 + \\ & 2 \cdot Y_{MUX} \cdot Y_{redun}(1 - Y_{redun})) \\ = & Y_{non-redun} \cdot Y_{MUX} \cdot Y_{redun}(2 - Y_{redun}) \\ = & e^{-\lambda \cdot (A - A_{redun})} \cdot e^{-\lambda \cdot A_{MUX}} \cdot e^{-\lambda \cdot A_{redun}} \cdot \\ & (2 - e^{-\lambda \cdot A_{redun}}) \\ = & e^{-\lambda \cdot A} \cdot e^{-\lambda \cdot A_{MUX}} \cdot (2 - e^{-\lambda \cdot A_{redun}}) \\ \text{yield gain} = & e^{-\lambda \cdot A_{MUX}} \cdot (2 - e^{-\lambda \cdot A_{redun}}) \end{aligned} \quad (3)$$

Equation (3) shows that yield improvement from redundancy is a function of redundant logic and MUX areas (and independent of A). Figure 2 shows yield improvements for various A_{redun} and A_{MUX} with $\lambda = 10^{-6}$. Based on the figure, we can see that yield improvement increases with A_{redun}/A_{MUX} ratio and saturates for a fixed A_{redun}. This implies that achievable yield improvement is limited by A_{redun}. We also notice that A_{redun}/A_{MUX} ratio must be sufficiently large (e.g., larger than 2) to achieve noticeable yield improvement.

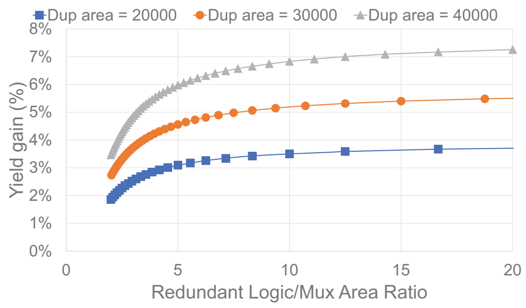

Fig. 2: Projected yield improvement by our model. Three different areas of redundant logic (cell area in redundant clusters) are projected in this simulation.

B. Yield Model

To connect the cluster-level yield calculation in Equation (1) with component-level (cells and nets) yield, we split the yield of a logic cluster into *back-end-of-line* (BEOL) and *front-end-of-line* (FEOL) yields. Therefore, the yield of a redundant cluster can be determined as given by Equation (5).

$$Y_{cluster} = Y_{BEOL}(nets) \cdot Y_{FEOL}(cells) \quad (4)$$

$$Y_{redun} = Y_{BEOL}(redun\_nets) \cdot Y_{FEOL}(redun\_cells) \quad (5)$$

We then estimate Y_{BEOL} and Y_{FEOL} in different ways. For Y_{FEOL}, we make assumption regarding the defect rate of the minimum NAND2 cell and then extrapolate defect rates of other cells based on the area ratio, as shown in Equation (6). The NAND2 defect rate (F_{nand}) is assumed to be $1ppm$ in our experiments. The overall FEOL yield is given by the product of all the cells, as shown in Equation (7).

$$\ln(Y_{cell}) = \ln((1 - F_{nand}) \cdot A_{nand2}/A_{cell}) \quad (6)$$

$$Y_{FEOL} = \prod_{j \in \text{all cells}} Y_{cell} \quad (7)$$

For the BEOL, we interpolate the yield based on the calibration of *Mentor Calibre*. We first characterize the yield of a routed design in a 28nm FDSOI foundry technology, and then use the yield data for curve fitting. We first run through the P&R flow to generate a GDSII file, then use *Mentor Calibre Yield Analyzer* [27] to estimate the BEOL yield for each $10um$ by $10um$ grid. The probability distribution of defect sizes is obtained from the *Calibre* reference flow. We use six metal layers in this yield characterization and other experiments. Since the pitches are the same for each layer, we use the same BEOL yield model for each of six layers. The yield of BEOL is given by the following equations.

$$Y_{BEOL} = \prod_{i \in \text{all grids}} Y_i, \quad (8)$$

$$where\ Y_i = e^{-\lambda(U_i)} \quad (9)$$

The Poisson exponent is assumed to be a function of track utilization U_i, and the data points of exponent $\lambda(U_i)$ are given by the *Calibre* result. Based on the curve fitting against *Calibre* data, we use $a = 3.91 \times 10^{-7}$ and $b = 2.9 \times 10^{-9}$ in our yield evaluation.

$$\lambda(U_i) = a \cdot U_i \ + \ b, \quad \text{where } i \text{ is the grid index} \tag{10}$$

We obtain the Y_{BEOL} in Equation (9) by decomposing the design into grids, calculating U_i of each grid, and using the fitting equation in Equation (10) to derive Y_i. We need per-net yield for design yield calculation in Equation (5) after adding redundancy. The per-net yield Y_{net} is then derived from the yield Y_{BEOL} by the following equations.

$$Y_{BEOL} = \prod_{\text{all nets}} Y_{net} \tag{11}$$

$$Y_{net} = e^{\frac{ln(Y_{BEOL})}{TOTAL\_WL} \cdot NET\_WL} \tag{12}$$

III. Methodology for Redundant Logic Insertion

Analysis in Section II shows that to maximize yield gain, we need to maximize area of redundant logic with minimized MUX area overhead. Our redundant logic insertion methodology addresses this through two optimization steps. First, we identify high-quality candidate logic clusters with large ratio of logic area to MUX area. Second, we select clusters using an ILP solver to maximize redundant area without undue timing degradation.[2] Details of these optimization steps are described in the rest of this section.

A. Two-Way FM-Based Cluster Generation

Previous works have proposed Rent-based netlist clustering to generate high-quality logic clusters [13] [22] [12]. The Rent-based netlist clustering reduces the number of terminals of a given cluster to save routing resources in the P&R flow. Inspired by Rent-based netlist clustering, we propose to use recursive min-cut bipartitioning to generate candidate clusters.[3] We use *MLPart* [4], a well-known implementation of multilevel two-way Fiduccia-Mattheyses (FM) [9].

Recursive bipartition is implemented based on the infrastructure of *RentCon* [28], which recursively calls *MLPart*, to bipartition clusters as long as a lower bound constraint on cluster size (numbers of cells in a cluster), denoted by S_{min} satisfied. An area balance constraint of 10% is imposed in each call to bipartitioning. *RentCon's* top-down application of *MLPart* identifies a given netlist's *intrinsic* (partitioning-based) Rent parameter (i.e., the lowest-slope trace possible in the plot of $\log(T)$ (y-axis) versus $\log(C)$ (x-axis), where C is the size of a cluster of logic gates and T is the associated number of terminals (cut nets at the boundary of the cluster)). See Figure 4 of [12]. We avoid using large S_{min} values because a large cluster is more difficult to be duplicated due to placement and routing congestions, as well as utilization constraints. Also, a larger cluster is more likely to degrade circuit timing during logic redundancy insertion. On the other hand, a small S_{min} is likely to produce small clusters which incur proportionally larger area overheads when replicated. Therefore, we fix S_{min} to 200 ($\sim$2% of smaller testcase (*AES*)) in this work.

B. ILP-based Cluster Selection

The quality of identified logic cluster during the redundancy insertion has a significant impact on P&R quality after ECO placement and route. We use MUX cells, which not only occupy available whitespace but also increase the risk of worse post-ECO timing, to switch between logic clusters. To reduce the overhead, our methodology first identifies low-overhead logic clusters and then selects compatible clusters which do not hurt the existing critical paths. Details will be described in the rest of this section.

In order to avoid timing impact on the design, we apply an ILP-based cluster selection flow to duplicate a maximum amount of logic under given timing constraints. The formulation uses binary variables to denote if a selected cluster will impact any timing-critical path due to the MUX cells associated with the cluster. The details of our ILP formulation are described as follows.

Maximize

$$\sum_{k=1}^{K} A_k \cdot C_k \tag{13}$$

Subject to:

$$\text{For } 1 \le n \le N, \quad SL_n - \sum_{m=1}^{M_n} P_{m,n} \cdot D_{MUX} \ge SL_{min} \tag{14}$$

$$\text{For } 1 \le n \le N, \ 1 \le m \le M_n, \ 1 \le k \le K \quad P_{m,n} \ge C_k, \tag{15}$$

where C_k and $P_{m,n}$ are binary variables,

$$(\sum_{k=1}^{K} A_k \cdot C_k + A_{init})/A_{chip} \le D \tag{16}$$

Our objective in Equation (13) guides the ILP to maximize the area of redundant logic clusters. To avoid the inserted MUX cells from hurting critical timing paths, we extract timing paths with less than 200ps slack from P&R tool and use Equation (14) and (15) to account for the timing slack after inserting redundancy. To prevent the inserted redundancy from excessively using available whitespace, we use Equation (16) to constraint the target die utilization after redundancy insertion.

IV. Experiments and Results

We implement our flow with *C++* code and commercial physical implementation tools [26] [29]. We solve the ILP formulation by calling *IBM ILOG CPLEX*[4] and feed our solutions to commercial tools. We use a 28nm FDSOI foundry technology in our experiments. The three testcases *AES*, *LEON3*, and *NETCARD* are obtained from OpenCores [30] and ISPD contests [31].

The main results are presented in Table II. Since the initial layouts constrained at lower frequencies and initial utilizations show better yield improvement, we report details of these data points to show the design tradeoffs under different settings. For redundant logic insertion, we target 10%, 20%, and 30% of the total core area. Thus, for each design, we show metrics targeting 60%, 70%, and 80% utilizations.

The limited yield gain of *AES* is caused by the following reasons. First, achievable redundant area is a strong function of original design area. *AES* has an original logic area of $84400um^2$, which is much smaller than *LEON3* and *NETCARD*. Based on analysis in Figure 2, we do not expect any significant yield gain for such a small design. For *LEON3* (*70*$\times$ size of *AES*), the yield gain is noticeably larger. For the same testcase, we observe that the yield improvement increases when more redundant logic cells are inserted. For example, the yield improvements of *LEON3* are 1.20$\times$, 1.41$\times$ and 1.62$\times$ when target utilizations after redundancy insertion are 60%, 70% and 80%, respectively. We observe timing degradation right after ECO place and route, and the timing slacks are restored after optimization, which is indicated by near-zero *worst negative slack* (WNS) and *total negative slack* (TNS) in *LEON3*. *NETCARD* also shows high yield improvement because of larger design area, but we notice that DRC violations occur in *NETCARD* since its physical implementation is wire-dominanted.

For all testcases, the power, leakage, and routed wirelength increase with higher amount of redundancy. We observe high power overhead in the post-ECO results, due to the logic redundancy including flip-flops. In addition, we propagate switching activity (we use an activity factor of 0.02) from output pins of flip-flops in vectorless power estimation; this makes the flip-flop power overhead more obvious. Note that the timing of *AES* is tight and hence the ILP is infeasible at high clock frequency. We use "–" in those rows.

[2]We use ILP to obtain optimum cluster selection solutions, and use only critical timing paths to help maintain tractable problem scales.

[3]Inadequate cluster choice can ruin the redundancy insertion. An extreme example is to choose a single NAND2 cell as the cluster. The area overhead to use a MUX cell for function selection will be high.

[4]The runtimes of the two-way partitioning and ILP are less than 178 minutes for *LEON3* (440K gate count) on a Xeon E5-2690 machine.

TABLE II: Yield improvement and attributes of (1) original routed layouts (Orig), (2) post-ECO layouts with redundant logic (ECO), and (3) post-optimization layouts (Opt).

Design (ns)	Clock (ns)	Init. Util.	Target Util.		#Insts	Area (μm^2)	Final Util.	Yield	WNS (ps)	TNS (ns)	P_{tot} (mW)	P_{leak} (mW)	WL (μm)	#DRC	#MUX	Redundant Cells	MUX Area	Redundant Area (μm^2)
AES	2.0	50%	60%	Orig	11487	8440	50.1%	98.3%	176	0.001	1.7	0.18	150243	44	0	0	0	0
				ECO	1.23x	1.19x	1.19x	1.00x	-504	-12.941	1.12x	1.83x	1.44x	103	55	2559	161	1422
				Opt	1.23x	1.19x	1.19x	1.00x	0	0.001	1.12x	1.94x	1.44x	93	55	2559	161	1430
AES	2.0	50%	70%	Orig	11487	8440	50.1%	98.3%	176	0.001	1.7	0.18	150243	44	0	0	0	0
				ECO	1.42x	1.36x	1.36x	1.00x	-549	-17.111	1.24x	2.83x	1.74x	102	125	4743	367	2685
				Opt	1.42x	1.36x	1.37x	1.00x	-8	-0.011	1.29x	2.89x	1.74x	67	125	4743	367	2692
AES	2.0	50%	80%	Orig	11487	8440	50.1%	98.3%	176	0.001	1.7	0.18	150243	44	0	0	0	0
				ECO	1.57x	1.50x	1.50x	1.01x	-990	-66.251	1.47x	4.00x	2.01x	171	221	6280	649	3608
				Opt	1.57x	1.51x	1.51x	1.01x	-21	-0.051	1.47x	4.22x	2.01x	183	221	6280	649	3616
AES	2.0	50%	90%	Orig	11487	8440	50.1%	98.3%	176	0.001	1.7	0.18	150243	44	0	0	0	0
				ECO	1.67x	1.66x	1.67x	1.01x	-1538	-134.501	1.82x	5.94x	2.30x	1000	380	7360	1116	4494
				Opt	1.68x	1.67x	1.67x	1.01x	3	0.001	1.88x	6.56x	2.30x	1000	380	7360	1117	4504
AES	0.8	50%	90%	Orig	11942	10242	60.8%	97.9%	-5	-0.011	6.1	1.74	146107	65	0	0	0	0
				ECO	–	–	–	–	–	–	–	–	–	–	–	–	–	–
				Opt	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AES	0.8	80%	90%	Orig	11719	9538	90.5%	98.1%	-17	-0.161	5.8	1.5	120016	77	0	0	0	0
				ECO	–	–	–	–	–	–	–	–	–	–	–	–	–	–
				Opt	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LEON3	4	50%	60%	Orig	442613	621758	50.4%	28.1%	-13	-0.042	147.9	17.63	9562023	2	0	0	0	0
				ECO	1.17x	1.18x	1.18x	1.21x	-1213	-10854.600	1.20x	1.46x	1.21x	0	3456	71694	10153	102006
				Opt	1.17x	1.18x	1.18x	1.20x	-18	-0.070	1.20x	1.47x	1.21x	1	3456	71694	10153	102029
LEON3	4	50%	70%	Orig	442613	621758	50.4%	28.1%	-13	-0.042	147.9	17.63	9562023	2	0	0	0	0
				ECO	1.33x	1.35x	1.35x	1.41x	-1585	-30674.900	1.42x	2.09x	1.42x	7	8350	136489	24532	194484
				Opt	1.33x	1.36x	1.36x	1.41x	-26	-0.082	1.43x	2.14x	1.42x	7	8350	136489	24533	194523
LEON3	4	50%	80%	Orig	442613	621758	50.4%	28.1%	-13	-0.042	147.9	17.63	9562023	2	0	0	0	0
				ECO	1.47x	1.51x	1.51x	1.63x	-1694	-48297.300	1.67x	2.77x	1.66x	0	13324	196155	39145	279319
				Opt	1.48x	1.52x	1.52x	1.62x	-31	-0.091	1.68x	2.87x	1.66x	1	13324	196155	39145	279361
NETCARD	4	50%	60%	Orig	303000	399021	50.2%	44.1%	39	0.000	98.1	12.84	12381761	90	0	0	0	0
				ECO	1.15x	1.17x	1.17x	1.09x	-2276	-755.229	1.24x	1.67x	1.19x	1000	3870	41472	11370	54595
				Opt	1.15x	1.17x	1.17x	1.09x	-22	-0.038	1.24x	1.71x	1.19x	1000	3870	41472	11377	54725
NETCARD	4	50%	70%	Orig	303000	399021	50.2%	44.1%	39	0.000	98.1	12.84	12381761	90	0	0	0	0
				ECO	1.29x	1.33x	1.33x	1.18x	-3468	-5453.600	1.49x	2.46x	1.38x	1000	8546	79095	25108	104952
				Opt	1.29x	1.33x	1.33x	1.17x	-234	-1.311	1.51x	2.61x	1.38x	1000	8546	79095	25151	105227
NETCARD	4	50%	80%	Orig	303000	399021	50.2%	39.1%	39	0.000	98.1	12.84	12381761	90	0	0	0	0
				ECO	1.42x	1.48x	1.48x	1.43x	-3506	-18876.000	1.76x	3.33x	1.56x	1000	13709	114146	40277	152329
				Opt	1.43x	1.50x	1.50x	1.41x	-1590	-30.580	1.81x	3.68x	1.57x	1000	13709	114146	40436	152620

V. Conclusion

Yield is now a dominant challenge in a new technology node, and yield loss during early learning stages of a new process can make leading-edge product chips economically unviable. To mitigate yield loss due to random defects, we propose a redundant logic insertion methodology that copies clusters of logic cells and connects their outputs (i.e., fanouts) to original nets through MUX cells. Based on a Poisson yield model, we derive *yield gain* as a function of redundancy cells and MUX areas. We show that maximum achievable yield gain is determined by redundant cell area.

Our methodology optimizes redundant logic insertion through two optimization steps. First, we extract candidate clusters with minimum cuts through a recursive bipartitioning algorithm. Second, we maximize area of redundant logic by selecting the best clusters via solution of an integer-linear program. Experimental results on our benchmark circuits show that for large design areas, logic redundancy can improve defect-limited yield by up to 1.62$\times$ from an initial value of 28.1%. Such a yield improvement could be highly significant especially for products in a new technology node, where profit margins are large due to the lack of competition.

Although our study focuses on defect-limited yield, the concept of opportunistic redundant logic insertion, as well as our methodology, can be applied toward other purposes such as (i) improvement of product lifetime against aging through redundancy, (ii) mitigation of impact of random soft defects on chip performance, etc. Our ongoing and future works include early stage routability consideration, timing recovery, and exploration of algorithms to improve the quality of clusters for purposes of redundancy insertion for yield gain. For example, the use of top-down multilevel FM bipartitioning might be adapted to incorporate elements of classic "replication cut" approaches [14] [17]. Another potential direction is to select optimum clusters from a richer set that is derived using multiple runs of partitioning. Intelligent heuristics to choose minimum cluster size for different designs may also improve the current approach.

References

[1] C. Bamji and E. Malavasi, "Enhanced Network Flow Algorithm for Yield Optimization", *Proc. DAC*, 1996, pp. 746-751.

[2] Y. Bourai and C.-J. R. Shi, "Layout Compaction for Yield Optimization via Critical Area Minimization", *Proc. DATE*, 2000, pp. 122-127.

[3] M. T. Buehler, J. M. Cohn, D. J. Hathaway, J. D. Hibbeler and J. Koehl, "Use of Redundant Routes to Increase the Yield and Reliability of a VLSI Layout", US patent US7308669 B2, 2007.

[4] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Improved Algorithms for Hypergraph Bipartitioning", *Proc. ASP-DAC*, 2000, pp. 661-666.

[5] H.-Y. Chen, M.-F. Chiang, Y.-W. Chang, L. Chen and B. Han, "Full-Chip Routing Considering Double-Via Insertion", *IEEE TCAD* 27(5) (2008), pp. 844-857.

[6] A. Ceyhan, M. Jung, S. Panth, S. K. Lim and A. Naeemi, "Impact of Size Effects in Local Interconnects for Future Technology Nodes: A Study Based on Full-Chip Layouts", *Proc. ITC/AMC*, 2014, pp. 345-348.

[7] V. K. R. Chiluvuri and I. Koren, "Layout-Synthesis Techniques for Yield Enhancement", *IEEE Trans. on Semiconductor Manufacturing* 8(2) (1995), pp. 178-187.

[8] M. Cho, H. Xiang, R. Puri and D. Z. Pan, "TROY: Track Router with Yield-Driven Wire Planning", *Proc. DAC*, 2007, pp. 55-58.

[9] C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", *Proc. DAC*, 1982, pp. 175-181.

[10] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner and T. Mudge, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation", *Proc. Micro*, 2003, pp. 7-18.

[11] R. S. Ghaida, K. Doniger and P. Zarkesh-Ha, "Random Yield Prediction Based on a Stochastic Layout Sensitivity Model", *IEEE Trans. on Semiconductor Manufacturing* 22(3) (2009), pp. 329-337.

[12] L. Hagen, A. B. Kahng, F. J. Kurdahi and C. Ramachandran, "On the Intrinsic Rent Parameter and Spectra-Based Partitioning Methodologies", *IEEE TCAD* 13(1) (1994), pp. 27-37.

[13] B. Hu and M. Marek-Sadowska, "Congestion Minimization During Placement without Estimation", *Proc. ICCAD*, 2002, pp. 739-745.

[14] J. Hwang and A. El-Gamal, "Optimal Replication for Min-Cut Partitioning", *Proc. ICCAD*, 1992, pp. 432-435.

[15] T. Iizuka, M. Ikeda and K. Asada, "Timing-Driven Cell Layout De-Compaction for Yield Optimization by Critical Area Minimization", *Proc. DATE*, 2006, pp. 1-6.

[16] A. B. Kahng, B. Liu and I. I. Măndoiu, "Nontree Routing for Reliability and Yield Improvement", *IEEE TCAD* 23(1) (2004), pp. 148-156.

[17] C. Kring and A. R. Newton, "A Cell-Replicating Approach to Min-Cut Based Circuit Partitioning", *Proc. ICCAD*, 1991, pp. 2-5.

[18] S.-Y. Kuo, "YOR: A Yield-Optimizing Routing Algorithm by Minimizing Critical Areas and Vias", *IEEE TCAD* 12(9) (1993), pp. 1303-1311.

[19] Y.-W. Lee, Y.-H. Lin and Y.-L. Li, "Minimizing Critical Area on Gridless Wire Ordering, Sizing and Spacing", *J. of Information Science and Engineering* 30(1) (2014), pp. 157-177.

[20] K.-Y. Lee, C.-K. Koh, T.-C. Wang and K.-Y. Chao, "Fast and Optimal Redundant Via Insertion", *IEEE TCAD* 27(12) (2008), pp. 2197-2208.

[21] A. Nardi and A. L. Sangiovanni-Vincentelli, "Synthesis for Manufacturability: a Sanity Check", *Proc. DATE*, 2004, pp. 796-801.

[22] T. K. Ng, J. Oldfield and V. Pitchumani, "Improvements of a Mincut Partition Algorithm", *Proc. ICCAD*, 1987, pp. 470-473.

[23] E. Papadopoulou, "Net-aware Critical Area Extraction for Opens in VLSI Circuits via Higher-Order Voronoi Diagrams", *IEEE TCAD* 20(5) (2011), pp. 583-597.

[24] Z. Wo, I. Koren and M. Ciesielski, "An ILP Formulation for Yield-driven Architectural Synthesis", *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2005, pp. 12-20.

[25] IBM ILOG CPLEX. http://www.ilog.com/products/cplex/

[26] Cadence Innovus User Guide.

[27] Mentor Calibre User's Manual.

[28] RentCon. http://vlsicad.ucsd.edu/WLD/RentCon.pdf

[29] Synopsys Design Compiler User's Manual.

[30] OpenCores. http://opencores.org

[31] ISPD Contest Website. http://www.ispd.cc/contests/