DiCER: Distributed and Cost-Effective Redundancy for Variation Tolerance

Di Wu*, Ganesh Venkataraman†, Jiang Hu†, Quiyang Li† and Rabi Mahapatra*
†Dept. of Electrical Engineering, Texas A&M University, College Station, TX 77843, U.S.A
{ganesh,jianghu,qiuyang}@ee.tamu.edu
*Dept. of Computer Science, Texas A&M University, College Station, TX 77843, U.S.A
dwu@tamu.edu, rabi@cs.tamu.edu

Abstract— **Increasingly prominent variational effects impose imminent threat to the progress of VLSI technology. This work explores redundancy, which is a well-known fault tolerance technique, for variation tolerance. It is observed that delay variability can be reduced by making redundant paths distributed or less correlated. Based on this observation, a gate splitting methodology is proposed for achieving distributed redundancy. We show how to avoid short circuit and estimate delay in dual-driver nets which are caused by gate splitting. A spin-off gate placement heuristic is developed to minimize redundancy cost. Monte Carlo simulation results on benchmark circuits show that our method can improve timing yield from 59% to 72% with only 0.3% increase on cell area and 2.2% increase on wirelength on average.**

I. Introduction

When VLSI technology approaches nanoscale regime, circuit performance is increasingly affected by variational effects such as process variations, power supply noise, coupling noise, and temperature changes. In order to have sufficient safety margins for the variations, circuit designers have to set unnecessarily aggressive timing targets which may waste both design effort and power. On the other hand, current circuit designs are progressively limited by power budget and unnecessary waste on power is no longer tolerable. Therefore, timing variations need to be minimized together with critical path delays during timing optimization. Recently, several statistical gate sizing methods [1]–[4] are proposed for process variation aware timing optimization.

In this work, we explore another direction - hardware redundancy - for improving timing tolerance to variations. Hardware redundancy is a well known technology for fault tolerant systems [5]. In circuit designs, redundant transistors can help to reduce the chance of stuck-open faults [6]. Mesh [7] and redundant connections [8] have already been utilized to reduce clock skew variability. In contrast, redundancy has rarely been mentioned for signal path timing variation tolerance until the recent work of [9]. In [9], Triple Modular Redundancy(TMR), which is a classic fault tolerance technique, is applied in re-synthesis for variation tolerance [9]. However, this work is limited to pre-layout designs and does not consider interconnect delay which is a widely-recognized dominating factor for circuit timing.

In general, redundancy is a relatively expensive technique. For example, TMR requires two replicates of the original module plus a voting circuit and therefore at least triples the hardware cost and power consumption. Although redundancy is suitable for fault tolerance, it is often an unaffordable overkill for variation tolerance. In [9], substantial effort is made to modify the TMR technique so that the hardware replication is minimized. Nevertheless, the re-synthesis method in [9] still causes 20% increase on cell area for 10% improvement on delay variation tolerance.

[1]This work is partially supported by SRC under contract 2003-TJ-1124.

In this work, we propose a new variation tolerance driven redundancy methodology through gate splitting. Our major contributions are listed as follows.

- We reveal the relationship between spatial correlation and delay variability through an Elmore delay based analysis and SPICE based Monte Carlo simulations. The result shows that less correlation in redundancy may lead to less delay variability.
- According to the above observation, we propose a distributed redundancy technique based on gate splitting.
- We show how to avoid short circuit in dual-driver nets which are caused by gate splitting. We also developed an Elmore-like delay estimation method for dual driver nets.
- A spin-off gate placement heuristic is developed to minimize the cost of the distributed redundancy. In general, the cost of our method is significantly less than that of [9].
- The proposed redundancy methodology is designed for post-placement optimization, therefore, the corresponding timing results are more meaningful than that of [9].

Please note that our approach is radically different from traditional gate duplication works [10]–[12] even though they have some superficial resemblance. The spin-off gate and the original gate drive the same set of nets in our approach while in traditional gate duplication works [10]–[12] the duplicated gate and the original gate drive separated nets and no signal redundancy exists. The proposed technique can be applied together with statistical gate sizing [1], [2] to further improve timing yield which is the probability of satisfying timing constraints considering variations.

Monte Carlo simulations are performed on benchmark circuits after placement legalization by commercial tool. Spatially correlated process variations and power supply variations are considered in the simulations. The results show that our method can improve timing yield from 59% to 72% with only 0.3% increase on cell area and 2.2% increase on wirelength on average.

II. Motivation Examples

In this section, we will use a few examples to demonstrate that delay variability depends on the correlation between redundant paths. These examples, which include both Elmore delay based analysis and SPICE based Monte Carlo simulations, motivate us to pursue distributed redundancy for delay variability reduction.

A. Elmore Delay Based Analysis

The first example is simply a 1-sink net as shown in Figure 1(a). The wire is split into two halves which are routed separately as in Figure 1(b). Please note that the wire area in (b) is the same as in (a) as the wire width in (b) is a half of that in (a). We will show that the variability of the delay from the driver to the sink is less in (b) than in (a) through an Elmore delay based analysis.

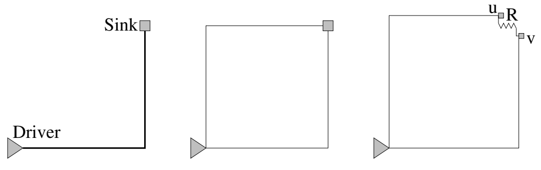

Fig. 1. A simple example of redundancy.

Obviously, Figure 1(b) is a non-tree routing and the delay can be obtained by the method in [13]. More specifically, a non-tree can be partitioned into a tree plus a set of link edges. The Elmore delay in the tree can be calculated easily. Then, the non-tree delay can be obtained by successively adding the link edges and updating the delay. The non-tree in Figure 1(b) can be partitioned into two parts connected by a virtual link resistor $R = 0$ as in Figure 1(c). In this partitioning, the sink capacitance is split into two equal parts u and v which are at the same location even though they are drawn separately in Figure 1(c). If the Elmore delays without link R are t_u and t_v, respectively, the delays after inserting link $R = 0$ become [13]:

$$\hat{t}_u = t_u - \frac{t_u - t_v}{r_u - r_v} r_u \quad (1) \qquad \hat{t}_v = t_v - \frac{t_u - t_v}{r_u - r_v} r_v \quad (2)$$

where r_u and r_v are equal to the Elmore delay at u and v, respectively, when node capacitance $C_u = 1, C_v = -1$ and the other node capacitances are zero [13]. By defining $\alpha = \frac{r_u}{r_u - r_v}$, the above equations can be rewritten as:

$$\hat{t} = \hat{t}_u = \hat{t}_v = (1-\alpha)t_u + \alpha t_v \quad (3)$$

It is not difficult to see that $r_v < 0$ and therefore $0 < \alpha < 1$.

Observation 1: ***In an RC circuit, a short (link resistor of $R = 0$) between two redundant paths averages the delays of the two paths.***

Since Equation (1) and (2) hold for any arbitrary non-tree, *observation 1* is true for general RC circuits although it is derived based on the example of Figure 1 for the simplicity of illustration.

If variations are considered, delay t_u and delay t_v become two random variables with standard deviation σ_u and σ_v, respectively. Let the covariance between t_u and t_v be $\sigma_{u,v}$. Since the delay $\hat{t}$ is a linear combination of t_u and t_v, the variance of delay $\hat{t}$ is given by:

$$\hat{\sigma}^2 = (1-\alpha)^2\sigma_u^2 + \alpha^2\sigma_v^2 + 2\alpha(1-\alpha)\sigma_{u,v} \quad (4)$$

If α is approximated as a constant, we can reach the following conclusion which is an important basis of our work.

Observation 2: ***The delay variance of shorted redundant paths in an RC circuit increases(decreases) as the covariance among the redundant paths increases(decreases).***

The rationale behind *observation 2* or Equation (4) is that less correlated delay variations along redundant paths have more chances to cancel out each other in the delay averaging implied by *observation 1*. *Observation 2* does not depend on any specific variation model and is applicable for many different kinds of variations such as process variation, power supply noise and temperature change.

Since the correlation between two redundant path delays normally increases when they are moved close to each other [14], the layout in Figure 1(a) can be treated as aggregated redundant paths with perfect correlation. Thus, $\hat{\sigma}^2$ has the maximal value in Figure 1(a) when covariance $\sigma_{u,v}$ reaches its maximal value of $\sigma_u\sigma_v$. On the other hand, separating redundant paths apart as in Figure 1(b) can reduce spatial correlation between the two path delays and thereby reduce the variability of the averaged delay $\hat{t}$. This observation implies that gate/wire sizing [1], [2] improves timing yield more through reducing nominal delays than through reducing delay variability. Therefore, making redundancy to be distributed may reduce delay variability further than performing gate/wire sizing alone. This is very similar as the diversified redundancy design in fault tolerant systems [15].

B. SPICE Based Monte Carlo Simulations

Since the Elmore delay model is sometimes inaccurate despite its high fidelity, we perform SPICE based Monte Carlo simulations to verify *observation 2*. To be more general, buffers are included in the interconnect. Buffer gate length variation and wire width variation are considered and assumed to follow Gaussian distribution [14]. A layout area is tessellated into an array of tiles indicated by the dashed grid in Figure 2. The spatial correlations among the variations are handled by applying the Principle Component Analysis (PCA) method on this grid as in [14]. The variations in the same tile are approximately treated with perfect spatial correlation. Two variations at two tiles far apart have relatively small spatial correlation.

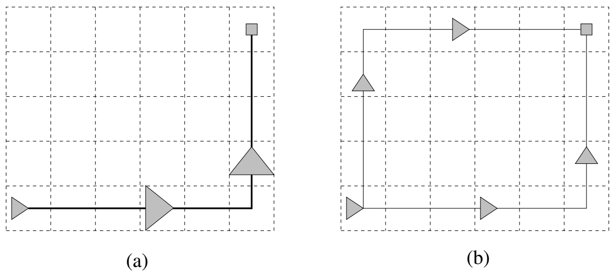

Fig. 2. Layout configurations for Monte Carlo simulations.

In the SPICE simulation, a single wire path with two buffers in Figure 2(a) is split into two wire paths and four buffers as in Figure 2(b). The buffer and wire size in Figure 2(b) is a half of that in Figure 2(a). Therefore, there is no buffer or wire area change due to the splitting. The *distance* between the two redundant paths in Figure 2(b) is specified in term of the number of tiles in rectilinear space. For example, the distance between the two paths in Figure 2(b) is 6. We vary the distance and observe the impact to delay variations at the sink.

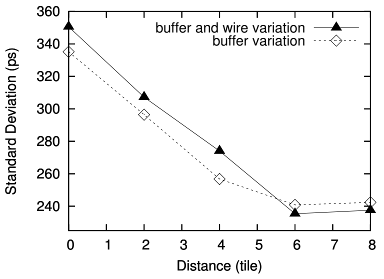

Fig. 3. Standard deviations of delay vs. distance between split buffers.

The Monte Carlo simulation results are depicted in Figure 3. The horizontal axis indicates the distance between two redundant paths. The zero distance results are obtained before the splitting. As the distance varies, the change on the mean value of delays is negligible. Therefore, only the delay variations in term of standard deviation are shown in Figure 3. The lower curve is from the result considering only buffer gate length variations while both the gate length variation and the wire width variation are included for the upper curve. These curves show a common trend that the delay variation reduces when the redundant paths are far apart and the spatial correlation is reduced. These are supporting evidence for *Observation 2* based on accurate gate and wire models.

III. Gate Splitting Methodology

Based on *observation 2*, we propose a gate splitting technique for reducing signal delay variability. The basic idea of gate splitting is illustrated in Figure 4. A gate G_0 on the timing critical path is split into two halves and the **spin-off gate** G_f is placed some distance away from G_0 with separated wire connections indicated by the dashed lines.

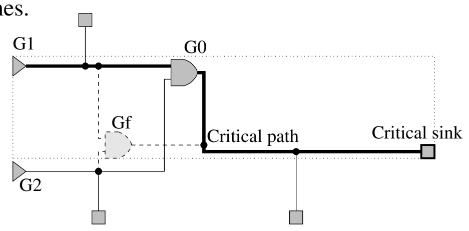

Fig. 4. An example of gate splitting. G_f is the spin-off gate and the dotted rectangle is the feasible region.

If gate G_0 originally has size w_0, its size after splitting and the size of spin-off gate G_f are $w_0/2$. If there is no gate of size $w_0/2$ in cell library, they are rounded to the closest available size in library. Alternatively, we can perform *gate duplication* where G_0 retains its original size and the size of gate G_f also equals w_0. However, the gate duplication here is quite different from traditional gate duplication works [10]–[12] in which the duplicated gates drive separated nets and no signal redundancy exists. Therefore, we use the term of *gate splitting* to avoid confusion. In fact, the sizes of G_0 and G_f can be tuned in a post processing for either gate splitting or gate duplication.

Another issue worth discussion is wiring cost. Even though it seems that layout in Figure 2(b) has the same wire area as that in Figure 2(a), the former consumes more wire pitch than the later. In modern IC designs, wire resource is more scarce than gate resource. In addition, separated wires may encounter more crosstalk noise. Therefore, we emphasize more on gate splitting than wire splitting and try to minimize extra wires incurred by gate splitting.

The gate splitting can be performed after either cell placement or global routing. Without loss of generality, we assume that the input for gate splitting includes a cell placement solution and Steiner trees for each signal net. We propose a gate splitting methodology which is composed by the following four phases.

1) **Gate selection**. In this phase, the gates to be split are identified. There are two options for the selection. One approach is to select gates based on their timing criticality and variability. For example, gates along paths with large disutility function value [1] can be selected. The other approach is to select gates according to a gate sizing solution [1], [2], [16] as gate sizing algorithms are mostly decided by timing criticality [1], [2], [16] and/or timing variability [1], [2]. Usually the gates along timing critical paths are sized up. The gates which are sized up in sizing can be regarded as aggregated redundancy and can be split to obtain distributed redundancy.
2) **Spin-off gate placement**. The spin-off gates are placed such that they are far apart from the original gates, the monotonicity of each critical path is not degraded, short circuit in dual-driver nets is avoided and the wiring cost is minimized. This phase will be discussed in details in Section V.
3) **ECO placement and placement legalization**. The spin-off gate placement may cause cell overlaps which need to be removed through ECO(Engineering Change Order) placement or placement legalization. There are existing ECO placer and placement legalizer tools which can be adopted directly.
4) **Spin-off gate connection**. The spin-off gates need to be connected to their fanin and fanout nets through wires. For each fanin net, an extra sink due to the spin-off gate is added and a new Steiner tree [17] can be constructed to accommodate this change. For the fanout net, the spin-off gate make it become a dual-driver net. Since there is no Steiner tree algorithm for dual-driver nets, to the best of our knowledge, we just connect the spin-off gate with the original fanout Steiner tree through the shortest feasible routes considering wiring blockages and congestions. If the gate splitting is performed after global routing, then ECO routing needs to be conducted.

IV. Handling Dual-driver Nets

Due to the gate splitting, a net may be driven by two drivers as in Figure 4. This phenomenon raises two issues which do not exist in conventional single driver routings. One is the risk of short circuit between the two drivers. The other is the fast estimation of signal delays in dual-driver nets. We have developed a dual-driver delay model considering both issues. For details, please refer to [18].

V. Spin-off Gate Placement

A. Problem Formulation

The spin-off gates (like G_f in Figure 4) need to be placed such that they are far apart from the original gates, the monotonicity of each critical path is not degraded, short circuit in dual-driver nets is avoided and the wiring cost is minimized. The short circuit avoidance constraint has been introduced in Section IV. The other objectives and constraints will be described as follows.

Monotonicity is one of the most desired properties for timing critical paths. For a path $a \rightsquigarrow b \rightsquigarrow c$ which starts from node a and then reaches node c through node b, it is monotone if $length(a \rightsquigarrow b) + length(b \rightsquigarrow c) = length(a \rightsquigarrow c)$. If node b is outside of the minimum bounding box containing a and c, path $a \rightsquigarrow b \rightsquigarrow c$ is not monotone. Obviously, a monotone path can achieve better timing than a non-monotone path. This fact motivates gate duplication works [11], [12] to straighten non-monotone critical paths for timing improvement.

In our work, we wish to reduce delay variability without hurting the nominal delay. Therefore, we need to ensure that the spin-off gate placement does not degrade monotonicity of each critical path. This can be achieved by restricting spin-off gates within the feasible region [12] of the critical paths it is associated with. The concept of **feasible region** is suggested in [12] and we restate it here for the completeness of the description. Consider splitting a gate G_0 with fanin gates $\mathcal{G}_{in} = \{G_1, G_2, ..., G_m\}$. Let $\mathcal{G}_c \subseteq \mathcal{G}_{in}$ be the set of fanin gates on timing critical paths and $v_{0,c}$ be the most critical sink in the fanout net of G_0. The feasible region $R_f(G_j, G_0, v_{0,c})$ for a fanin gate $G_j \in \mathcal{G}_c$ is simply the minimum bounding box containing G_j, G_0 and $v_{0,c}$. The feasible region $R_f(\mathcal{G}_c, G_0, v_{0,c})$ for the entire set of critical fanin gates is the overlap of the feasible regions for all gates in $\mathcal{G}_c$. As an example shown in Figure 4, the dotted bounding box is the feasible region and G_1 is the only fanin gate of G_f on timing critical paths.

A major objective of spin-off gate placement is to separate it far apart from the original gate so that the spatial correlation between them as well as the delay variability can be reduced. Thus, the distance between the original gate G_0 and the spin-off gate G_f is $d(G_0, G_f)$ needs to be maximized in the spin-off gate placement.

Last but not the least, the wiring cost needs to be minimized. Since expensive cost is a major hurdle that prevents wide application of redundancy for variation tolerance, cost minimization is one of our major goals. As gate and wire size can be tuned in a post processing, we focus on wirelength minimization in the spin-off gate placement.

The wiring cost is determined by the wire connection between the spin-off gate G_f and T_0, which is the fanout Steiner tree driven by G_0, and the set of fanin Steiner trees $\mathcal{T} = \{T_1, T_2, ..., T_m\}$ for gate G_0. In order to quantify the wiring cost, we need to define the distance $d(G_f, T_i)$ between gate G_f and a Steiner tree $T_i \in \{T_0, T_1, ..., T_m\}$. This is based on defining the distance $d(G_f, e_{uv})$ between G_f and an edge $e_{uv} \in T_i$ with two end nodes u and v. The distance $d(G_f, e_{uv})$ is equal to the distance between the location (x_f, y_f) of G_f and the closest connection point (x_c, y_c) which is given by:

$$x_c = \text{median}(x_f, x_u, x_v) \qquad\qquad y_c = \text{median}(y_f, y_u, y_v)$$

where (x_u, y_u) and (x_v, y_v) are the coordinates of node $u \in T_i$ and node $v \in T_i$, respectively. Thus, the distance between G_f and edge e_{uv} is:

$$d(G_f, e_{uv}) = \sqrt{(x_f - x_c)^2 + (y_f - y_c)^2} \tag{5}$$

Then, the distance between gate G_f and Steiner tree T_i is defined as

$$d(G_f, T_i) = \min_{\forall e_{uv} \in T_i} d(G_f, e_{uv}) \tag{6}$$

The above objectives and constraints can be summarized as a formulation of the spin-off gate placement problem as:

$$\mathcal{SGP}:$$

$$\begin{aligned} &\text{Minimize} && \textstyle\sum_{i=0}^{m} \omega_i d^2(G_f, T_i) - \lambda d^2(G_f, G_0) && (7)\\ &\text{Subject to} && G_f \in R_f(\mathcal{G}_c, G_0, v_{0,c}) && (8)\\ & && \min(\tau_{0 \rightsquigarrow f}, \tau_{f \rightsquigarrow 0}) > \beta \Delta_{0f,max} && (9) \end{aligned}$$

where ω_i and λ are non-negative weighting factors. The first term in the objective function (7) is for wiring cost minimization and the second term is to separate the spin-off gate G_f apart from the original gate G_0. The reason that we employ the quadratic objective function in (7) is the same as conventional quadratic placement [19]. The weighting factor ω_i is decided based on the timing criticality. We adopt the timing criticality definition in [20]. The maximum combinational logic path delay from primary input (or flip-flop) to primary output (or flip-flop) is denoted as D_{max} and the timing slack at gate G_i is represented as $slack(G_i)$. Then, the timing criticality or the net weight can be obtained as [20]:

$$\omega_i = (1 - \frac{slack(G_i)}{D_{max}})^{\gamma} \tag{10}$$

where γ is a constant. The constraint (8) is to restrict the spin-off gate within the feasible region so that there is degradation on the monotonicity of critical paths. The constraint (9) states that the lower bound of the delay $\tau_{0 \rightsquigarrow f}$ ($\tau_{f \rightsquigarrow 0}$) from G_0 (G_f) to G_f (G_0) is at least $\beta (> 1)$ times the maximal difference $\Delta_{0f,max}$ between the signal arrival time at G_0 and G_f so that the short circuit risk is avoided [18].

B. Algorithm

The spin-off gate placement problem $\mathcal{SGP}$ formulated above is difficult to be solved directly because of: (a) The constraint (9) is troublesome. Depending on the location (x_f, y_f) of the spin-off gate, the connection point on the fanout Steiner tree T_0 may change from one edge to another edge. Thus, $\tau_{0 \rightsquigarrow f}$ and $\tau_{f \rightsquigarrow 0}$ are not continuous functions with respect to the location (x_f, y_f). (b) The distance function $d(G_f, T_i)$ in the objective function (7) is not well-behaved as the connection point $v_{i,c}$ varies depending on location (x_f, y_f).

We employ two common and effective techniques - *relaxation* and *restriction* - to handle the difficulties on solving $\mathcal{SGP}$. The relaxation is applied to constraint (9), i.e., constraint (9) is temporarily dropped and any violation on it will be fixed later. The restriction technique is applied to $d(G_f, T_i)$ in the objective function (7). More specifically, we temporarily restrict the connection point for tree T_i at a node $v_{i,c} \in T_i$. Then, we can first solve the following relaxed and restricted problem. The method of fixing violations on (9) and selection of connection nodes $v_{i,c}$ will be described later.

$$\mathcal{SGP}_{\mathcal{RR}}:$$

$$\begin{aligned} &\text{Minimize} && \textstyle\sum_{i=0}^{m} \omega_i d^2(G_f, v_{i,c}) - \lambda d^2(G_f, G_0) && (11)\\ &\text{Subject to} && G_f \in R_f(\mathcal{G}_c, G_0, v_{0,c}) && (12) \end{aligned}$$

Problem $\mathcal{SGP}_{\mathcal{RR}}$ is a constrained quadratic programming problem and can be solved along x and y directions separately as in quadratic placement [19]. The feasible region $R_f(\mathcal{G}_c, G_0, v_{0,c})$ is normally a rectangle and can be represented by its corner coordinates $(x_{min}, y_{min}) - (x_{max}, y_{max})$. The location of gate G_0 is at (x_0, y_0). The location of a connection node $v_{i,c}$ is represented by $(x_{i,c}, y_{i,c})$. Then, the subproblem along x direction becomes:

$$\begin{aligned} &\text{Minimize} && \phi(x_f) = \textstyle\sum_{i=0}^{m} \omega_i (x_f - x_{i,c})^2 - \lambda (x_f - x_0)^2\\ &\text{Subject to} && x_{min} \le x_f \le x_{max} \end{aligned}$$

If $\tilde{x}_f$ satisfies $\frac{d\phi(x_f)}{dx_f}|_{x_f = \tilde{x}_f} = 0$ and $\hat{x}_f = \min(x_{max}, \max(x_{min}, \tilde{x}_f))$, the above problem has an optimal solution at:

$$x_f^* = \begin{cases} x_{min} & : \frac{d^2\phi(x_f)}{dx_f^2} \le 0, \phi(x_{min}) \le \phi(x_{max}) \\ x_{max} & : \frac{d^2\phi(x_f)}{dx_f^2} \le 0, \phi(x_{min}) > \phi(x_{max}) \\ \hat{x}_f & : \text{otherwise} \end{cases} \tag{13}$$

The optimal solution along the y direction can be obtained similarly.

```
Procedure: $SpinoffGatePlacement(G_0, T_0, \mathcal{T})$
Input: Gate $G_0$ to be split
        Fanout Steiner tree $T_0$, critical sink $v_{0,c} \in T_0$
        A set of fanin Steiner trees $\mathcal{T} = \{T_1, T_2, ...T_m\}$
Output: Location $(x_f, y_f)$ of spin-off gate $G_f$
1. Find feasible region $R_f(\mathcal{G}_c, G_0, v_{0,c})$
2. Initialize $\lambda$
3. $v_{i,c} \leftarrow$ centroid of $T_i$, $i = 0, 1, ..., m$
4. $(x_f, y_f) \leftarrow$ solve $\mathcal{SGP}_{\mathcal{RR}}$
5. While (true)
6.    $v_{i,c} \leftarrow$ closest connection on $T_i$ to $(x_f, y_f)$
          $i = 0, 1, ..., m$
7.    $(x_f, y_f) \leftarrow$ solve $\mathcal{SGP}_{\mathcal{RR}}$
8.    If inequality (9) is satisfied
          Return $(x_f, y_f)$
9.    Else if $(x_f, y_f)$ is at corner of $R_f(\mathcal{G}_c, G_0, v_{0,c})$
          Return failure
10. Else $\lambda = \lambda + \sigma$
```

Fig. 5. Main algorithm of spin-off gate placement.

The overall algorithm for solving the spin-off gate placement problem $\mathcal{SGP}$ is summarized in Figure 5. Initially, we approximate the connection points $v_{i,c}$ by the centroid of each tree T_i as indicated by step 3 of Figure 5. If a Steiner tree T_i has nodes $V_i = \{v_{i,0}, v_{i,1}, ...\}$ including the source, sinks and Steiner nodes, then the x coordinate of its centroid is simply the average value of x coordinates of all nodes in V_i. The y coordinate of the centroid can be obtained similarly. After locations of the connection points are found, an optimal solution of $\mathcal{SGP}_{\mathcal{RR}}$ can be obtained in step 4 as described above. After the initial solution is found in step 4, the solution is refined iteratively starting from step 5. At the beginning of each iteration which is step 6, the connection points $v_{i,c}$ are updated according to the (x_f, y_f) obtained previously. Then, the solution can be refined by running $\mathcal{SGP}_{\mathcal{RR}}$ at step 7. The constraint (9) is checked at step 8. If the solution satisfies constraint (9), the algorithm is finished. Otherwise, the value of λ is increased by a small positive value σ at step 10 and next iteration is started. Increasing the value of λ may push the spin-off gate G_f farther away from gate G_0 and may improve the chance that constraint (9) is satisfied. If gate G_f is pushed to a corner of the feasible region and constraint (9) is not satisfied yet, it is quite likely that there is no feasible solution and the iterations terminate at step 9. It is not hard to see that this iterative procedure is similar as Lagrangian relaxation and λ plays a role of Lagrangian multiplier.

TABLE I

EXPERIMENTAL RESULTS.

	Nominal delay (ps)		Std deviation (ps)		Timing yield			Cell area (μm^2)		Wirelength (μm)		$d(G_0, G_f)$		CPU
Circuit	Initial	Split	Initial	Split	Constraint	Initial	Split	Initial	Split	Initial	Split	ave(tile)	#split	(sec)
c432	6094	5633	327	239	6100	28.8%	94.8%	7154	7222	5897	6570	8.4	16	1.18
c880	4250	4255	214	154	4440	66.8%	77.0%	16203	16291	15458	15872	3.6	23	1.26
c1355	4044	4086	335	288	4550	60.4%	62.8%	21831	21870	18197	18634	3.6	24	1.65
c1908	5412	5354	280	295	5800	82.6%	90.3%	40280	40415	29250	29829	3.0	40	2.74
c2670	5135	5131	297	249	5350	48.0%	60.6%	52215	52391	76117	76659	2.0	32	2.96
c3540	7491	7450	597	560	8100	57.2%	65.0%	68915	69109	64800	65441	1.4	41	5.80
c5315	6789	6751	412	392	7000	35.0%	50.6%	99601	99835	133848	134640	1.4	48	6.12
c6288	18912	18913	1528	1486	20500	69.6%	70.8%	96113	96132	60319	60710	0.7	62	15.65
c7552	6595	6519	339	318	7000	67.8%	74.4%	143136	143311	161401	162182	2.0	42	8.12
Average	Decrease 1%		Decrease 11%			59%	72%	Increase 0.3%		Increase 2.2%				

VI. Experimental Results

Our methodology and algorithm are tested on the ISCAS85 benchmark circuits with 180nm technology. All timing related parameters of the standard cells are extracted from HSPICE. The nominal wire resistance and capacitance is $0.076\Omega/\mu m$ and $0.118 fF/\mu m$, respectively. The experiments are performed on a Linux platform with a 1.3 GHz Intel processor. The initial placement is obtained from Cadence Silicon Ensemble. Initial Steiner trees are constructed by using the C-Tree [17] software downloaded from GSRC Bookshelf(http://dropzone.tamu.edu/˜cnsze/GSRC/ctree.html). Phase 1 gate selection and phase 2 spin-off gate placement algorithms are implemented in C++ by ourselves. The ECO placement of phase 3 is also obtained from Cadence Silicon Ensemble. In phase 4, the Steiner trees for the fanin nets of spin-off gates are constructed by C-Tree. Spin-off gates are connected to their existing fanout Steiner tree through the shortest feasible route by our own implementation.

The final timing results are estimated through Monte Carlo simulations. Variations on gate length, power supply level and wire width are considered. These variations are assumed to follow Gaussian distribution with standard deviations equal to 10% of their nominal values. Spatial correlations among the variations are handled by the PCA(Principle Component Analysis) method as in [14]. In applying the PCA method, the die area of each circuit is tessellated into an 64 $\times$ 64 array of tiles.

To the best of our knowledge, there is no similar approach in previous works. Hence, comparisons are made between the initial results and the results after our 4-phase gate splitting method. The results are shown in Table I. The data of the maximal nominal path delay are in column 2 and 3. Same as the observation in Section II-B and our expectation, the influence from our method to the nominal delay is usually small and is only an average of 1% reduction.

The data in column 4 and 5 of Table I are the standard deviations of the maximal path delay variation. Except c1908, our method always reduces the standard deviations. The magnitude of the reduction is often large when the distance $d(G_0, G_f)$ between the original gate G_0 and the spin-off gate G_f is large. The average distances in term of the number of tiles are listed in column 13. For c432, the standard deviation is reduced by 27% from the gate splitting as the average distance is greater than 8 tiles. In contrast, the standard deviation reduction for c6288 is only 3% since its average distance is less than 1. Normally, the distance $d(G_0, G_f)$ is constrained by the size and aspect ratio of feasible regions. Overall, our method can reduce the standard deviation by 11% on average.

The timing yield results are shown in column 7 and 8 of Table I. These data are obtained based on the timing constraints in column 6. Occasionally, the improvement from our method is insignificant due to tight feasible region constraints like in circuit c6288. Otherwise, our method usually results in significantly better timing yield than the initial results. Our method can increase the timing yield from an average of 59% to an average of 72%.

The cost overhead of our method is very small in general. The increase on cell area is at a negligible level of 0.3% and the increase on total wirelength is 2.2% on average. Therefore, our approach is much more cost-effective than the work of [9] which increases cell area by 20%. The numbers of split gates are in column 14 of Table I. The total CPU time of our method is also displayed in Table I. Usually, our method takes only a few seconds to complete.

VII. Conclusion

In order to cope with the threat of variation problems, we explore redundancy technique which is a relatively new direction for variation tolerance. We show that distributed redundancy can effectively reduce delay variability. The redundancy cost and short circuit risk can be handled through a careful algorithm design for spin-off gate placement.

References

[1] S. Raj, S. B. K. Vrudhula, and J. Wang. A methodology to improve timing yield in the presence of process variations. *DAC*, pages 448–453, 2004.

[2] S. H. Choi, B. C. Paul, and K. Roy. Novel sizing algorithm for yield improvement under process variation in nanometer technology. *DAC*, pages 454–459, 2004.

[3] M. Mani, A. Devgan, and M. Orshansky. An efficient algorithm for statistical minimization of total power under timing yield constraints. *DAC*, pages 309–314, 2005

[4] J. Singh, V. Nookala, Z. Luo, and S. S. Sapatnekar. Robust gate sizing by geometric programming. *DAC*, pages 315–320, 2005.

[5] B. W. Johnson. *Design and analysis of fault-tolerant digital systems*. Addison-Wesley Publishing Company, 1989.

[6] N. Sirisantana, B. C. Paul, and K. Roy. Enhancing yield at the end of the technology roadmap. *IEEE Design and Test of Computers*, 21(6):563–571, Nov-Dec 2004.

[7] P. J. Restle, et al. A clock distribution network for microprocessors. *IEEE J. of Solid-State Circuits*, 36(5):792–799, May 2001.

[8] A. Rajaram, J. Hu, and R. Mahapatra. Reducing clock skew variability via cross links. *DAC*, pages 18–23, 2004.

[9] S.-C. Chang, C.-T. Hsieh, and K.-C. Wu. Re-synthesis for delay variation tolerance. *DAC*, pages 814–819, 2004.

[10] A. Srivastava, R. Kastner, C. Chen, and M. Sarrafzadeh. Timing driven gate duplication. *IEEE TVLSI*, 12(1):42–51, January 2004.

[11] G. Beraudo and J. Lillis. Timing optimization of FPGA placement by logic replication. *DAC*, pages 196–201, 2003.

[12] G. Chen and J. Cong. Simultaneous timing-driven placement and duplication. *ISFPGA*, pages 51–59, 2005.

[13] P. K. Chan and K. Karplus. Computing signal delay in general RC networks by tree/link partitioning. *IEEE TCAD*, 9(8):898–902, August 1990.

[14] H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. *ICCAD*, pages 621–625, 2003.

[15] S. Mitra, et al. A design diversity metric and analysis of redundant systems. *IEEE Trans. on Computers*, 51(5):498–510, May 2002.

[16] C.-P. Chen, C. C.-N. Chu, and D. F. Wong. Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation. *IEEE TCAD*, 18(7):1014–1025, July 1999.

[17] C.J. Alpert, G. Gandham, M. Hrkic, J. Hu, A.B. Kahng, J. Lillis, B. Liu, S.T. Quay, S.S. Sapatnekar, and A.J. Sullivan. Buffered Steiner trees for difficult instances. *IEEE TCAD*, 21(1):3–14, January 2002.

[18] G. Venkataraman, N. Jayakumar, J. Hu, P. Li, S. Khatri, A. Rajaram, P. McGuinness and C. Alpert. Practical link insertion to reduce skew and its variations in buffered clock networks. *ICCAD*, 2005.

[19] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE TCAD*, 10(3):356–365, March 1991.

[20] A. Marquardt, V. Betz, and J. Rose. Timing-driven placement for FPGAs. *ISFPGA*, pages 203–213, 2000.