STATISTICAL GENERIC AND CHIP-SPECIFIC SKEW ASSIGNMENT FOR IMPROVING TIMING YIELD OF FPGAS

Satish Sivaswamy and Kia Bazargan

Electrical Engineering Dept.
University of Minnesota, MN 55455
{satish,kia}@umn.edu

ABSTRACT

This paper presents a technique to fix timing violations caused by process variations in FPGAs by adjusting the clock skews of flip-flops. This involves making the clock distribution network tunable by adding programmable delay elements to compensate for variations. We propose generic as well as chip-specific skew assignment schemes that are robust to variations. The two proposed schemes result in recovering about 80% and 82% of the failed chips respectively with conservative timing constraints. With more aggressive constraints, the corresponding numbers are 69% and 77% respectively. Our technique causes a 39% increase in the number of chips in the *fast* bin when speed-binning is performed. The area and power overhead associated with this technique are 3.5% and 5.6% respectively.

1. INTRODUCTION

With feature sizes shrinking well below the 90nm regime, process variations are causing a significant number of chips to fail the timing/power specifications. This causes a large number of chips to be either completely discarded or sold at a lower price with speed binning. To alleviate the problems due to variability, statistical optimization techniques have been adopted for a long time by ASIC designers. Recently the FPGA community has also started to take note of the potential pitfalls associated with not considering process variations in the design phase([1], [2], [3]). Statistical static timing analysis has been widely used to account for variations and to produce tighter delay distributions to improve timing yield. Timing yield of sequential circuits can be further increased by tuning skews on the clock network. Clock skews can be adjusted either pre-silicon(based on timing slack distribution) or post-silicon to fix paths that violate timing constraints. The only requirement of such a scheme is the ability to tune the clock network.

The clock network can be tuned by inserting programmable delay elements(PDE) in the clock tree to distribute skews to various FFs to reduce timing violations in the circuit. FPGAs are unique in that they contain pre-fabricated elements with an option of programming the logic elements and interconnects. When compared to an ASIC, FPGAs have lower flexibility for the locations of PDEs to tune the clock skews since they need to be application independent. FPGA clock networks are also not typically designed with an H-tree topology since it is difficult to mesh it with a tile based layout. They are typically designed with a spine-and-ribs topology. As a result of these two restrictions, techniques developed to tune clock networks for ASICs cannot be directly applied to FPGAs. However, the advantage with FPGAs is that unintentional clock-skew due to variations is not as critical as ASICs since FPGAs typically run at much lower frequencies and variation in clock skew is a small component of the cycle time. There has been prior work in clock skew scheduling for FPGAs. In [4], the authors used several global clocks to deliver skews to various FFs. This approach incurs substantial power and routing overhead. In [5], the authors proposed a new clock-routing architecture consisting of a global H-tree and a local spine-and-ribs structure. They inserted PDEs on alternate branch points of the global H-trees to distribute skews. Though an H-tree based skew distribution scheme provides more useful skew between FFs connected to different branches of the tree, it leads to difficulty in layout. Modern FPGAs have clock de-skew elements that can be used to adjust the phase of clocks but they are limited in number and do not provide fine control of skews to different FFs.

In this work we propose a flat skew distribution scheme with little modification to the existing clock tree. We perform extensive architecture exploration to identify the number of delay blocks needed in each PDE to balance yield improvement with the area and power overhead of the delay elements. We also provide generic(common for all fabricated chips) and chip-specific techniques to fix timing violations and conduct some studies to see how these techniques fare with conservative(to achieve higher yield) as well as aggressive(to achieve higher performance) guard-banding of circuit elements. We further consider the impact of skew assignment on speed binning of chips.

The rest of the paper is organized as follows. Section 2 presents some modeling details. Section 3 motivates this work. Section 4 provides an architectural background of clock distribution schemes used in commercial FPGAs and

presents our proposed architecture. It also provides details about the PDE that we used in this study. Section 4.2 presents our architecture exploration technique. Skew assignment problem is formally stated in Section 5. Sections 6.1 and 6.2 discusses generic and chip-specific skew assignment and their results. Section 7 discusses the overhead associated with the proposed techniques.

2. MODELING VARIATIONS AND DELAY

In this section we present our process variation and delay models. We use a 65nm predictive technology model[6] and consider variations in the transistor length (L_{eff}), width(W_{eff}) and interconnect width(W_{int}) and thickness(T_{int}). We model these parameters as spatially correlated gaussian distributions. We use the spatial correlation model presented in [7]. Due to lack of real foundry data, we use the process parameters presented in [3]. The total 3σ variations in L_{eff}, W_{eff}, W_{int} and T_{int} are approximately 15%, 10%, 15% and 6% respectively. We model process variation as

$$X_{total} = X_{inter} + X_{intra} \quad (1)$$

Where X_{inter} refers to the inter-die component and X_{intra} refers to the intra-die component of variation. X_{total} refers to the total variation where $X \in \{L_{eff}, W_{eff}, W_{int}, T_{int}\}$.

As in [8] we express the delay of a circuit element using a simple first order Taylor series expansion around the nominal point. Delay is expressed as:

$$d = d_0 + \sum_{\forall p_i} S_{i_0} \Delta p_i \quad (2)$$

where d_0 is the nominal delay of the circuit element and $S_i's$ are the delay sensitivities of different process parameters. We obtain the sensitivities from HSpice simulations. In equation 2, $p_i's$ denote the various process parameters. To make computations easier and to capture correlations better, the correlated p_i variables are converted to independent components by using principal component analysis. Delay is then expressed as the nominal delay compounded with the weighted sum of principal components.

3. PRELIMINARIES

As mentioned in Section 1, adding PDEs to the clock network can reduce the number of chips that violate timing constraints. However, adding PDEs before every FF is prohibitive due to the area and power overhead. We must be judicious in selecting the number and locations of PDEs. Before addressing these issues, we first performed a preliminary study on MCNC benchmarks to determine the feasibility of using a tunable clock architecture to fix timing violations. We show the results in Figure 1. To generate data for Figure 1 we placed and routed MCNC sequential benchmark circuits using VPR[9]. We guard-band the individual circuit elements with their 3σ[1] delays and take the guard-banded

[1] 3σ delays are obtained from SPICE simulations

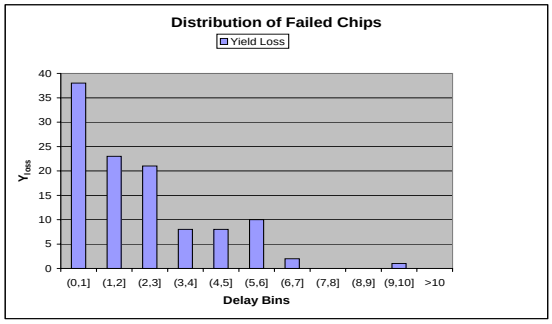
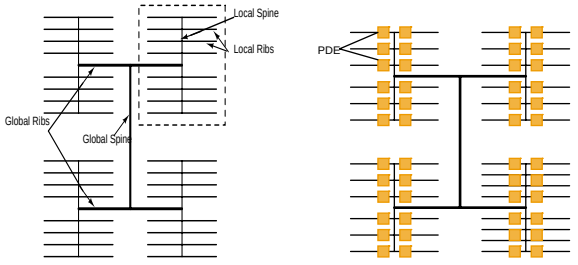


Fig. 1. Distribution of failed chips

delay (circuit delay when delay of all elements are bumped up by the guard-band factor) of circuits as the timing specification. Monte Carlo analysis with 10,000 samples is then performed to determine which chips violate the timing constraint. We then bin the chips into different delay bins based on the magnitude of timing violation. For instance, if a chip has a critical path delay that exceeds the specification by 1.4%, it is placed in the [1,2) delay bin meaning that the clock period of the chip exceeds timing specification by 1-2%. From the figure we can see that most of the failing chips fall in the bins (0,1] and (1,2]. Furthermore, about 88% of failed chips fall in the extended delay bin (0,5]. This indicates that we can correct a lot of timing violations if we can reduce the delay on critical paths of circuits by even a small amount.

4. CLOCK ARCHITECTURE

Clock distribution schemes in commercial FPGAs typically use a spine-and-ribs topology in which clocks are delivered to global spines, which then distributes them to various regions of the chip using the rib routing structures. This is illustrated in Figure 2(a). We propose to insert PDEs on the



(a) Spine and Rib topology (b) Tunable Clock Architecture

Fig. 2. Proposed clock Network

local ribs. This way, we insert 4 PDEs in each row giving us potentially 4 different skew values for every row. This is presented in Figure 2(b). We consider only one clock domain when presenting our approach for the ease of illustration. It can be easily extended to multiple clock domains. It is to be noted that all the logic clusters that derive their clocks from a local rib would have the same skew so we should take care

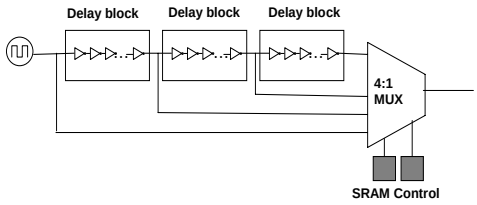


Fig. 3. Programmable Delay Element

when we assign skews to avoid causing new timing violations. Though having only 4 skew values per row does not give us very fine control of individual skews, it is still effective. This is because, in practice almost all timing violations occur between FFs that are spaced far apart giving us some margin to adjust the skews. From Figure 1 it is clear that we can recover a lot of failed chips with even a small margin. In the next section we describe the PDEs that we used in this study.

4.1. Programmable Delay Elements

We borrow the programmable delay element used in this work from [5]. It is shown in Figure 3. The clock signal passes through several delay blocks and the required skew is configured by storing appropriate values in the SRAM cells that control the output multiplexer. The delay blocks consist of a chain of 20 inverters each. It is to be noted that the available skews are discrete and depend on the delay of an individual delay block. We conducted SPICE simulations for a predictive 65nm technology and observed that the delay of a single delay block was around 150ps. In the next section we present our studies to determine the optimum number of delay blocks needed in every PDE to fix most of the timing violations.

4.2. Architecture Exploration

The number of chips failing the timing specification that can be recovered depends on the amount of useful skew that is available to fix the timing violations. This in turn depends on the number of delay blocks present in each PDE. We want to minimize the number of delay blocks to reduce the area and power overhead. We conducted a study to determine the optimum number of delay blocks needed in every PDE to balance the number of chips that can be recovered and the overhead associated with it. We observed that the number of additional chips that could be recovered increased substantially initially with the addition of PDEs but the gains started to diminish as we increased the number of delay blocks on each PDE beyond 3. As a result, we decided to use 3 delay blocks in our PDE to balance the effectiveness of the technique with its overhead. Similar experiments can be conducted to determine the ideal number of delay blocks incase each delay block contains a different number of inverters. In the next section we formally state the skew assignment problem and deal with layout constraints imposed by FPGAs and the discrete nature of available skews.

5. SKEW ASSIGNMENT PROBLEM

We present the skew assignment problem in this section with the help of Figure 4 from [10]. The figure shows a part of a sequential circuit containing a combinational block and a pair of flip-flops (FF_i and FF_j) at the input and output respectively. The skews at the input and output FFs are s_i and s_j respectively. The maximum delay of the combinational

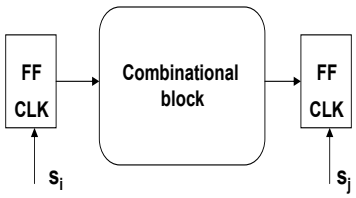


Fig. 4. Setup for writing timing constraints

block is denoted by $d_{max}(i,j)$ and the minimum delay by $d_{min}(i,j)$. T_{setup} and T_{hold} denote the setup and hold times of the FFs. Let T_{clock} be the clock period. With these notations, we have two sets of constraints on the slowest and fastest combinational paths between the two FFs. They are long and short path constraints and are written as [10]:

- Long Path constraints :

$$s_i + d_{max}(i,j) \leq s_j + T_{clock} - T_{setup}$$

- Short Path constraints :

$$s_i + d_{min}(i,j) \geq s_j + T_{hold}$$

The above constraints are computed for every pair of FFs that are connected by a combinational path in the circuit.

5.1. Layout and Discrete Skew Constraints

The skew assignment problem presented earlier is unconstrained since it does not take into account the layout constraints imposed by FPGAs. The formulation assumes that we have continuous control of skews of all FFs and all skews are independent. Neither of these assumptions are valid because

1. From Figure 2(b) it is clear that we have only 4 distinct skew values per row and hence skews to all FFs cannot be independently set.
2. Based on the number of delay blocks used in the PDEs, we have discrete levels of skew available for every PDE.

To account for layout constraints, we first cluster FFs that are driven by the same PDE in every row in the circuit. If we have an $n \times n$ FPGA and 4 PDEs per row, we will have a maximum of $4n$ clusters. All FFs belonging to a cluster have the same skew. To generate long and short path constraints, we proceed as in the unconstrained case by computing the maximum and minimum combinational delays between all pairs of FFs (i,j) connected by a combinational path. If i and j belong to different clusters, we

update(if necessary) the long and short path constraints between clusters. If i and j belong to the same cluster, we do not include the constraints. However, we still need to check if the critical path has a source and target FF belonging to the same cluster. We will not be able to reduce the critical path in this case due to layout constraints. In practice however, this is very rare since paths that are critical typically have their source and target spaced far apart and in our experiments we never encountered this problem.

Since we have only discrete levels of skew available, we introduce additional variables to account for the discrete skew problem. If there are L levels of skew available($S[1] \ldots S[L]$) then we introduce L additional (0,1) variables, $x_1, x_2, \ldots x_L$ for every skew variable i such that

$$\begin{aligned} x_{i1} + x_{i2} + \ldots + x_{iL} &= 1 \\ x_{i1} \cdot S[1] + x_{i2} \cdot S[2] + \ldots + x_{iL} \cdot S[L] &= s_i \end{aligned}$$

The above conditions ensure that skews obtained by solving the linear program satisfy the discrete skew constraints imposed by the PDEs.

6. ROBUST CLOCK SKEW ASSIGNMENT CONSIDERING PROCESS VARIATIONS

To fix timing violations caused by process variations, we should consider both the variability in the combinational sub-circuits as well as the clock distribution network itself. In this section we explore two possible solutions to robustly assign skews to different FF clusters to increase circuit tolerance to variability. The first is a generic technique used before a design is mapped to any particular FPGA and makes use of the estimated timing slack. The second technique is a chip-specific scheme that is used only for those chips that fail to meet timing specifications.

6.1. Generic Clock Skew Assignment

We first present a technique to robustly assign skews before the design is implemented on the FPGA. This is done after the place & route stage. We perform statistical static timing analysis(SSTA)[8] as a first step and compute $d_{max}(i,j)$ and $d_{min}(i,j)$ for every FF pair (i,j) that is connected by a combinational path. We use the statistical sum operation to compute the delay of a path and use a statistical max(min) operation to compute $d_{max}(i,j)$(d_{min})in the canonical form shown in equation 2. Let $(\mu_{max}, \sigma^2_{max})$ and $(\mu_{min}, \sigma^2_{min})$ be the (mean,variance) of the max and min delay distributions respectively. To perform skew assignment, we introduce a user controllable *uncertainty factor* k which increases variation tolerance by adjusting d_{max} and d_{min} based on the mean and variance of the distribution. We set

$$\begin{aligned} \overline{d}_{max}(i,j) &= \mu_{max} + k \cdot \sigma_{max} \\ \overline{d}_{min}(i,j) &= \mu_{min} - k \cdot \sigma_{min} \end{aligned} \quad (3)$$

and use $\overline{d}_{max}$ and $\overline{d}_{min}$ in the long and short path constraints respectively. Equation 3 increases only the robustness of the combinational paths. To account for the variations in the clock network, we introduce an explicit *robustness factor* R to the formulation. Based on the technique presented in [11], we first compute the statistical criticalities of all combinational paths $p \rightsquigarrow q$ between all FF pairs (p,q), where p and q belong to different FF clusters. We then use this criticality information ($crit(p,q)$) along with the robustness factor R to modify the problem formulation as follows

$$\begin{aligned} Maximize \quad & R \\ S.T. \quad \forall(i,j) & \\ s_i + \overline{d}_{min}(i,j) - crit(i,j)R &\geq s_j + T_{hold} \\ s_i + \overline{d}_{max}(i,j) + T_{setup} + crit(i,j)R &\leq s_j + T_{clock} \\ x_{i1} + x_{i2} + \ldots + x_{iL} &= 1 \\ x_{i1} \cdot S[1] + x_{i2} \cdot S[2] + \ldots + x_{iL} \cdot S[L] &= s_i \end{aligned} \quad (4)$$

The R factor in the above formulation increases the tolerance to variations in skew. The component $crit(i,j)$ is used to provide more variation tolerance to those clusters that are connected by paths having a high statistical criticality since those are the paths that are affected the most by variations in skew. We perform a binary search on T_{clock} to achieve the optimum clock period and maximize the robustness at each step by solving the linear program.

6.1.1. Experiments and Results

In this section, we first study the impact of skew assignment on speed binning of chips. Then we study the effectiveness of this technique in recovering failed chips with both conservative and aggressive timing constraints. Speed binning is performed by testing the operating frequency of every chip and classifying it into fast(T_{fast}), medium(T_{medium}), slow(T_{slow}) and failure bins. We place and route sequential MCNC benchmark circuits and perform skew assignment based on the formulation presented in Section 6.1. In this work we use deterministic place and route algorithms since our objective is to study the effectiveness of the skew assignment technique. However, skew assignment can be used in conjunction with statistical place and route algorithms ([1],[3]) to augment the timing yield further and to meet more aggressive timing specifications.

Using SSTA, we set the values of T_{fast},T_{medium} and T_{slow} such that fast, medium and slow bins contain 40%,30% and 29.9% of the chips respectively[1]. We then perform Monte Carlo simulations with 10,000 samples to gather statistics regarding the number of chips in each bin both with and without skew assignment. It is to be noted that for all experiments in this work, we consider variations in the skew elements also. The results obtained are shown in Figure 5. From the figure, it is clear that performing skew assignment is beneficial when speed binning is performed. The number of chips in the fast bin after skew assignment is 1.39X (up to 1.79X) the number without skew assignment. The numbers in the medium and the slow bins reduced to 0.88X and

Circuit	GB Factor = 3 σ			GB Factor = 2.5 σ		
	$T_{spec}(ps)$	#Fail	#Rec	$T_{spec}(ps)$	#Fail	#Rec
diffeq	10878.3	11	9	10468.4	50	40
tseng	6824.36	11	10	6549.06	40	26
s298	21756.5	13	10	20952.9	32	15
frisc	13848.5	15	12	13326.6	42	29
s15850	9969.61	8	8	9554.62	38	25
elliptic	15712.5	17	10	15014.1	52	30
s38417	13039.6	12	11	12574	50	37
bigkey	6877.99	10	8	6638.99	40	38
clma	26442.1	10	6	25497.94	30	18
dsip	4901.54	12	10	4731.28	43	37
Geo. Mn.		**11.65**	**9.25**		**41.08**	**28.19**

Table 1. Generic Skew Assignment Scheme

Circuit	GB Factor = 3σ		GB Factor = 2.5σ	
	# Fail	# Recov	# Fail	# Recov
diffeq	11	7	52	32
tseng	11	10	59	58
s298	6	4	51	37
frisc	12	11	48	40
s15850	9	9	34	27
elliptic	9	6	34	27
s38417	8	8	40	35
bigkey	12	11	64	56
clma	5	3	31	16
dsip	8	8	47	40
Geo Mn.	**8.77**	**7.15**	**44.77**	**34.67**

Table 2. Chip-Specific Skew Assignment Scheme

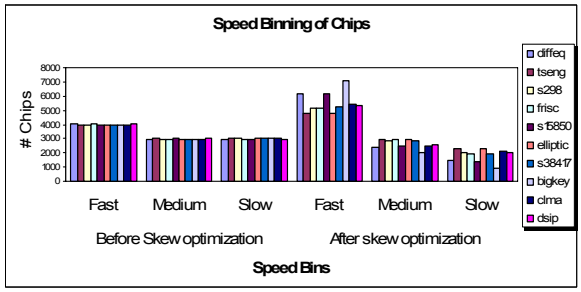


Fig. 5. Speed Binning

0.61X with skew assignment. Therefore performing skew assignment allows more chips to be sold as fast chips. For the 10 benchmark circuits and 100,000 total chips, 561 chips belonged to the failure bin without skew assignment. With skew assignment this number was 173. Skew assignment is also beneficial to recover more chips from the failure bin. Further, we observed that the mean of the delay distribution decreased but the standard deviation did not change much with skew assignment. We examined the benefits of skew assignment in more detail by considering both conservative and aggressive timing constraints.

We perform two sets of experiments to study the effectiveness of the technique in fixing timing violations. First, we place and route the circuit and use the 3σ guard-banded delay as the timing specification. The 3σ delay is a conservative specification. We then assign skews to different FF clusters based on the discussion in the previous section. Monte Carlo simulation with 10,000 samples is then performed to determine the number of chips that would have failed timing and the number saved with skew assignment. We then repeat the same experiment with a more aggressive timing constraint. For this, we chose the 2.5σ guard-banded delay as the specification. The results are shown in Table 1. The number of failing chips and those recovered are shown in the columns labeled #Fail and #Recov respectively. T_{spec} denotes the timing requirement to be met. Columns 3 and 4 are for the conservative case with 3σ cut-off delay and columns 6 and 7 are for the aggressive case with 2.5σ cut-off delay. From the table, we observe that our robust skew assignment scheme is effective in recovering chips that otherwise violate timing. It salvages about 80% of the failing chips with a conservative timing constraint and about 69% with an aggressive constraint. When we tighten the timing constraint, the percentage of chips recovered is marginally reduced. This can be attributed to the fact that the number of delay blocks per PDE, layout and discrete skew constraints, as well as the uncertainty in equation 3 all have bigger impact with a tighter timing constraint. This effect however, is only marginal and the scheme seems to perform well even with tight delay constraints.

6.2. Chip-Specific Skew Assignment

We presented a robust skew assignment scheme to be used after place & route but before the design is implemented on an FPGA. Since the technique is applied before implementation, there are uncertainties in delay estimation. In this section, we look at a chip-specific skew assignment scheme to fix timing violations.

The problem with such an approach is that we need access to the exact delays of all paths to successfully assign skews to different clusters. Clearly this is infeasible since we cannot measure the exact delays of all paths since there may be an exponential number of paths. However, we may be able to measure the delay of certain designated paths. To alleviate this problem, we divided the chip into 8 grids and instantiated a ring oscillator in each grid. This approach of using ring oscillators to capture correlations is similar to the technique in [12]. The idea here is to extract correlation information by measuring the operating frequency of the ring oscillators. Based on this measurement, we predict the delay of circuit elements. If the delay of the ring oscillator in a grid i, is T_{meas} and the delay distribution of the oscillator from SSTA is (T_μ, T_σ), then we introduce a *delay prediction factor*, p for each grid, calculated as

$$p_i = \frac{T_{meas} - T_\mu}{T_\sigma}$$

We compute $p_i's$ for all grids and use them to guard band all the circuit elements in that grid. For nets passing through multiple grids we use a weighted average of the $p_i's$. Figure 6 shows the normalized *actual* and the predicted critical path delay for benchmark circuits to demonstrate the accuracy of this technique. To measure the *actual* delay, we perform Monte Carlo simulation and this gives us access to the individual LUT and net delays. Using ring oscillators to predict delays captures the effects of correlations only to an extent and we did not expect it to be 100% accurate. The average error in prediction was about 1%. We can possibly bring it down further by instantiating more ring oscillators. However the accuracy we obtained is good enough for the purposes of this work. In order to perform skew assign-

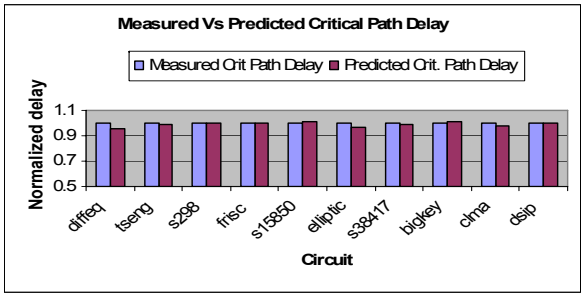


Fig. 6. Delay Prediction Accuracy

ment after design implementation, we first predict logic and net delays for failing chips. The skew formulation presented in (4) is then solved to get the optimum skews to fix timing violations. In the formulation, $\overline{d}$ is replaced by the predicted delays. The robustness factor is retained since we need to account for variations in the clock network. Once the skews are obtained, the design can be re-implemented on the chip and tested to verify timing.

6.2.1. Experiments and Results

As in Section 6.1.1 we perform experiments with conservative and aggressive timing constraints to verify the effectiveness of the chip-specific skew assignment scheme. We show the results in Table 2. The cut-off delays for these experiments are generated in the same way as in section 6.1.1. The chip-specific skew assignment scheme performs marginally better than the generic scheme. About 82% of the failing chips are recovered with conservative timing constraints and about 77% are recovered with more aggressive constraints. The reason this scheme performs slightly better is because the delay prediction reduces the uncertainty in the combinational blocks leading to a better skew assignment. This however, comes at the cost of increased test time. The generic scheme also has its advantages. It is better suited for speed-binning. If speed-binning is not used, the PDEs can be bypassed for the chips that do not require skew assignment, to reduce the power overhead.

7. OVERHEAD

The PDEs added to the clock tree to deliver programmable skews have an area and power overhead. Similar to VPR, we estimate the area overhead by counting the number of additional minimum width transistors needed. The PDE shown in Figure 3 consists of 150 transistors. If the FPGA chip contains 10,000 4-LUTs arranged in a 100×100 array, the area overhead in modifying the clock network as shown in Figure 2(b) is about 3.5%. We modified the clock architecture in *powermodel*[13] to compute the power overhead. Adding PDEs to the clock network increased the average power dissipation in the clock network by about 5.6%. We can reduce this power overhead by gating the clocks to the PDEs when they are not used. We can reduce this further by using energy efficient delay elements that work by increasing the switching resistance rather than the capacitance.

8. REFERENCES

[1] Y. Lin, M. Hutton, L. Hei, "Placement and Timing for FPGAs Considering Process Variations", *International Conference on Field Programmable Logic and Applications*, August 2006.

[2] Y. Lin, L. Hei, "Stochastic physical synthesis for FPGAs with pre-routing interconnect uncertainty and process variation" *International Sympossium on Field-Programmable Gate Arrays*,2007

[3] S. Sivaswamy, K. Bazargan, "Variation-Aware Routing for FPGAs", *International Sympossium on Field-Programmable Gate Arrays*, 2007.

[4] D.P. Singh, S.D. Brown, "Constrained Clock Shifting for field programmable gate arrays", *International Sympossium on Field-Programmable Gate Arrays*, 2002.

[5] Chao-Yang Yeh, Malgorzata Marek-Sadowska, "Skew-programmable Clock Design for FPGA and Skew-aware Placement", *Internation Sympossium on Field-Programmable Gate Arrays*, 2005.

[6] Berkeley Device Group, "Predictive Technology Model",in http://www.eas.asu.edu/simptm

[7] P. Froedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, C. Spanos, "Modeling Within-Die Spatial Correlation Effects for Process-Design Co-Optimization", *International Sympossium on Quality of Electronic Design* pp. 516-521. 2005.

[8] H.Chang, S. Sapatnekar, "Statistical Timing Analysis Under Spatial Correlations", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24 No. 9, pp. 1467-1482, Sep 2005.

[9] V. Betz, J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research", *International Conference of Field Programmable Logic and Applications*", 1997.

[10] Sachin Sapatnekar, "Timing", Springer-Verlag, New York, 2004.

[11] C.Visweswariah, K.Ravindran, K.Kalafala, S.G. Walker, S.Narayan, "First-Order Incremental Block-Based Statistical Timing Analysis", *Design Automation Conference*, 2004.

[12] P. Sedcole and P. Y. K. Cheung, "Within-die Delay Variability in 90nm FPGAs and Beyond", in Proc. *IEEE International Conference on Field Programmable Technology*, 2006.

[13] K.K.W. Poon, S.J.E. Wilton, A. Yan, "A Detailed Power Model for Field-Programmable Gate Arrays", *ACM Transactions on Design Automation of Electronic Systems*, Vol. 10, No. 2, pp. 279-302, April 2005.