TIMING VERIFICATION OF ADAPTIVE INTEGRATED CIRCUITS

A Thesis

by

ROHIT KUMAR

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2014

Major Subject: Computer Engineering

ABSTRACT

An adaptive circuit can perform built-in self-detection of timing variations and accordingly adjust itself to avoid timing violations. Compared with conventional over-design approach, adaptive circuit design is conceptually advantageous in terms of power-efficiency. Although the advantage has been witnessed in numerous previous works including test chips, adaptive design is far from being widely used in practice. A key reason is the lack of corresponding timing verification support. We developed new timing analysis techniques to fill this void. A main challenge is the large runtime complexity due to numerous adaptivity configurations. We propose several pruning and reduction techniques and apply them in conjunction with statistical static timing analysis (SSTA). The proposed method is validated on benchmark circuits including the recent ISPD'13 suite, which has circuit as large as $150K$ gates. The results show that our method can achieve orders of magnitude speed-up over Monte Carlo simulation with about the same accuracy. It is also several times faster than an exhaustive application of SSTA.

DEDICATION

To my parents

ACKNOWLEDGEMENTS

I would like to thank my advisor and committee chair, Dr. Jiang Hu, for his guidance and support throughout the course of this research. I would like to express my gratitude to Dr. Bing Li and Dr. Ulf schlichtmann from Technische Universität München, Germany for helping me with various issues and providing guidance during the course of this research. Besides my advisor I would like to thank my thesis committee members Dr. Peng Li and Dr. Duncan M. Walker for their encouragement and insightful comments.

I would also like to thank my friends Hao He, Jiafan Wang and Yiren Shen for helping me running the experiments and for many useful discussions we had which helped me to solve many problems concerning my work.

I would also like to thank my friends Nityendra Singh and Ayush Ruia for making my journey pleasant and motivating me throughout. I would like to express my gratitude to the department staff who helped me through various processes and enabled me to complete paperwork in a timely manner.

Finally, I would like to thank my family for all the encouragement and support during my graduate studies and research.

NOMENCLATURE

STA	Static Timing Analysis
SSTA	Statistical Static Timing Analysis
PCA	Principle Component Analysis
PERT	Program Evaluation and Review Technique

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

An adaptive circuit typically contains sensors to detect performance variations and autonomously compensates the variations by body biasing [26], supply voltage tuning [15], circuit reconfiguration [24], or other techniques. Unlike conventional over-design approach, which applies extra power uniformly (or blindly) across entire circuit (or all fabricated chips) to cushion variations, adaptive circuits apply power differentially at each block according to individually observed performance, i.e., in a targeted manner. Evidently, adaptive circuit design provides variation resilience in a more power-efficient manner than the over-design. This is especially true when the adaptivity is fine-grained (in blocks of hundreds/thousands of gates) and therefore has relatively precise compensation [15, 24]. Overall, adaptive circuit design simultaneously addresses two grand challenges faced by the VLSI design technology: variability and power.

Even with benefit demonstrated by test chips [26], adaptive circuit design is far from being widely adopted in realistic products. A main reason is the lack of corresponding timing verification support. The unconventional and sophisticated nature of adaptive design implies relatively large risk of design errors. Obviously, correct circuit functioning is more essential than potential power savings. As such, an adaptivity-aware timing analysis tool is a fundamental premise for wide application of adaptive circuit designs. One may consider to use conventional timing analysis on adaptive circuit with some simple tweaks. Such an approach is either very restrictive or inefficient. For example, conventional timing analysis based Monte Carlo simulation is directly applicable but very time consuming.

Another naïve approach is to apply conventional timing analysis to each adaptivity mode. This approach is practical only if the adaptivity is coarse-grained and the number of adaptivity blocks are small.

In this work, we attempt to find a general and practical approach to timing verification for adaptive circuit designs. Since adaptivity is operated according to observed variations in the delay of the critical paths and process variations are a dominant cause of the variations in the delay, so these have to be taken into consideration for timing analysis. Fortunately, statistical static timing analysis (SSTA) has been an active research subject [6,9,28] and had produced a rich body of techniques. Our study starts with an SSTA-based adaptivity enumeration approach. Then, we explore several pruning and reduction techniques in order to decrease computation cost. To our best knowledge, this is the first systematic effort on timing analysis for adaptive circuit design. The proposed techniques are tested on ISCAS'85 and ISPD'13 benchmark circuits, which include cases as large as $150K$ gates. Our technique is orders of magnitude faster than Monte Carlo simulation and provides very similar accuracy. Compared to SSTA-based exhaustive enumeration, our method yields almost identical result with several times less runtime.

1.1 Background

1.1.1 Process Variations

Process variations are the naturally occurring variation in the physical attributes of the transistors when the integrated circuits are fabricated. The variations occur in length, width, doping concentrations, gate oxide thickness and many other physical parameters of a transistor. These variations are dominant at smaller nodes such as nodes $< 65nm$ and the electrical performance of integrated circuits are impacted by these variations. These cause variations in the performance output of transistors and hence a logic gate.

The major variations occur in the length L_{eff} of the transistor. The other geometrical parameters such as oxide thickness, width etc. have a lower rate of change across the die. The wire geometry parameters such as width, height, thickness and resistivity also changes.

The parametric variations are often separated into tow categories : *inter-die* and *intra-die* [7]. The *inter-die* variation is the difference in the value of a parameter across nominally identical die and typically accounts for in circuit design as a shift in the mean of some parameter value equally across all devices or structures on any chip. The *intra-die* variation is the deviation occurring spatially within any one die. Such intra-die variation may have a variety of sources depending on the physics of the manufacturing steps that determine the parameter of interest. In contrast to inter-die variation, intra-die variations contributes to the loss of matched behaviour between structures on the same chip.

In modern devices the intra-die variations have significant effect on the performance parameters of the chip which were neglected earlier [19]. The increase in intra-chip parameter variations is due to the effects such as micro-loading in the etch, photo-resist thickness variations, optical proximity effect and stepper within-field aberrations as the manufacturing sizes approach the optical resolution limit [4]. Intra-die variations are spatially correlated: these are locally layout-dependent and are circuit specific i.e. devices with similar layout patterns and proximity structures tend to have similar characteristics.

1.1.2 Adaptive Circuits

An adaptive circuit as shown in Figure 1.1 has more than one supply voltage level, body-bias voltage levels or other kind of adaptivity configurations on which it can operate. It can switch to any available voltage level depending upon the timing failures in the circuit or can be specifically tuned post silicon to reduce the impact of the process variations. The purpose of applying the different levels is directed at two things, power and timing. The leakage power of a device has become a major issue in modern chip design with shrinking sizes of transistor. The amount of power lost in leakage current is significant and is comparable to the dynamic power of the chip. One of the technique to save this leakage power is to apply adaptive body bias to the chip. The application of reverse body bias reduces the leakage power by a large amount as the change in the leakage current changes exponentially with any change in the threshold voltage.

The other application of applying variable voltages either as body-bias or the supply voltage to the circuit is to compensate for any timing violation. A circuit can fail timing while in operation which could be due to many reasons such as ageing, process variations or other environmental reasons. This could lead to wrong operations of a circuit. Generally to meet the timing constraints of a chip which is failing, the supply voltage V_{dd} is changed but a combination of adaptive supply voltage, within-die adaptive body bias has been shown more effective in reducing the impacts of process variations as in [27].

For a circuit to function correctly the circuit has to meet the timing under every situation. For this purpose, to detect a failing timing path sensors such as Canary flip-flops or RAZOR [12] are generally applied on the critical paths which detect whether the path is failing timing or not and accordingly provides input to an adaptive policy which decides as which adaptive voltage level or adaptive body-bias should be applied to the circuit.

The uncertainty in timing due to the process variations in the circuit has become a major issue in the design of the circuits. As the transistor sizes are shrinking the effect of variations have become more and more significant. These variations give

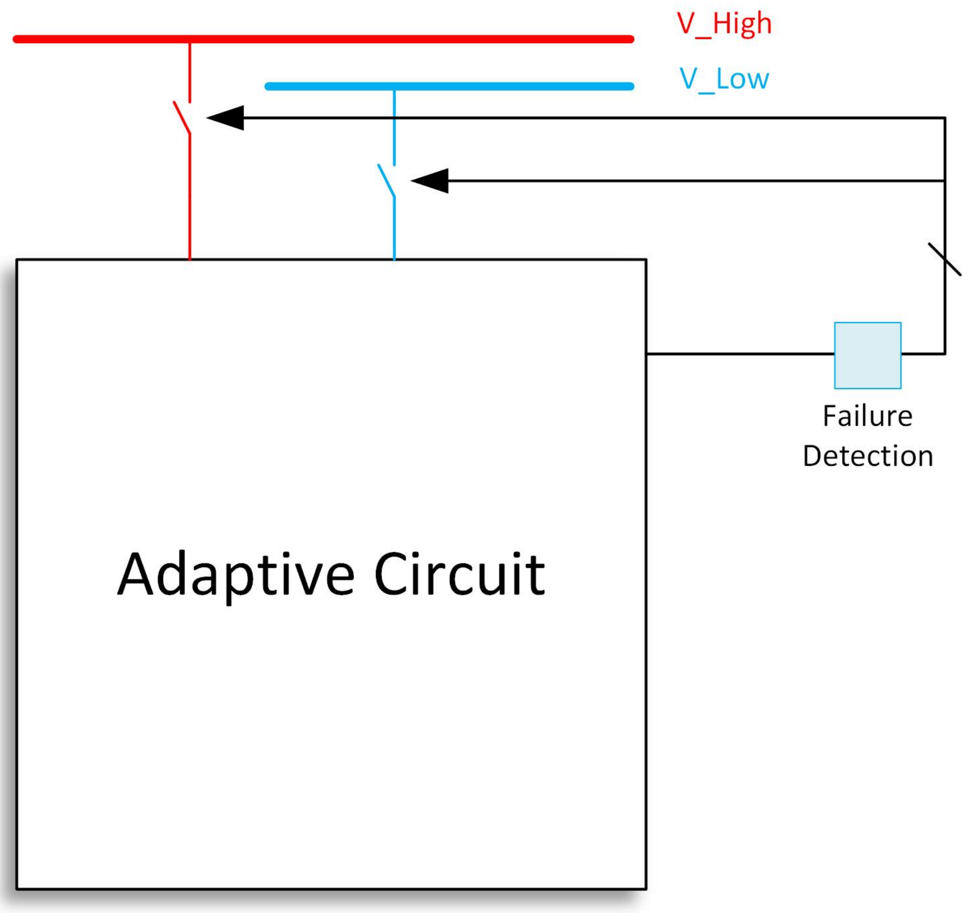
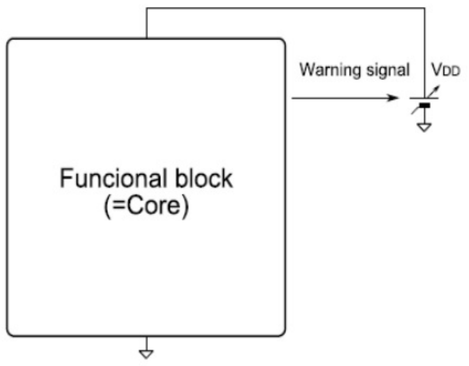


Figure 1.1: An adaptive circuit

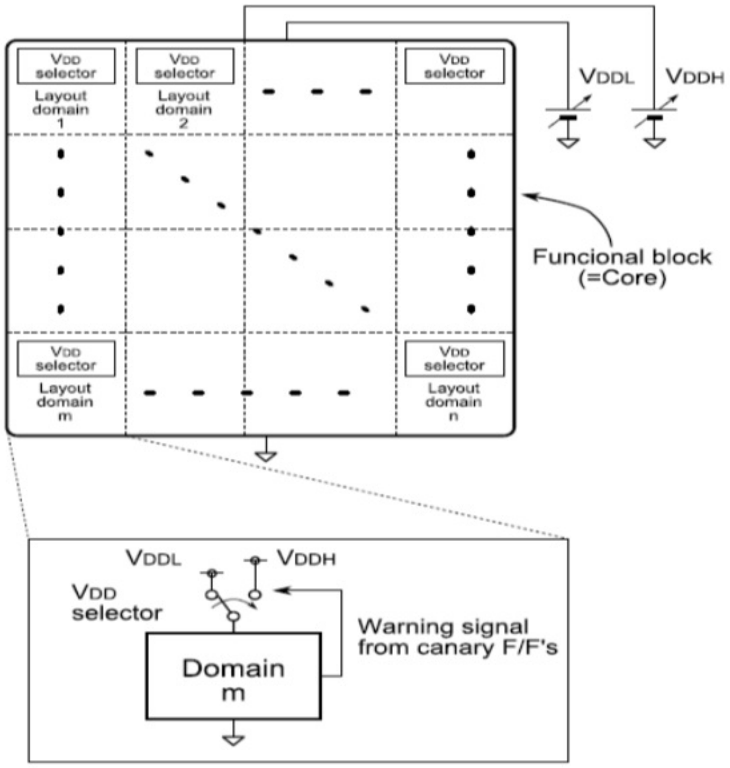
rise to uncertainty in the delay of the logic gates. Due to the variations the overall delay characteristics of the chip are no longer a deterministic value but a random value with a Gaussian distribution. As such with the conventional timing analysis tools the timing cannot be accurately predicted. Therefore a circuit may or may not be able to meet the timing as predicted. In such a case having adaptive voltage supply in the circuit becomes important for the correct operation of the circuit.

1.1.3 Fine Grained Adaptivity

The adaptivity can be applied to the whole circuit as a single unit i.e. in a coarse grained manner or it can be applied to the different parts of the circuit differently i.e. in a fine grained manner as shown in Figure 1.2. In fine grained adaptivity the whole circuit is divided into certain number of blocks/voltage islands. A single block will have a single voltage supply unit and this can operate at different voltage levels independent of other blocks. Each block can have its own sensors such as RAZOR circuit [12] or Canary flip-flops and a power supply unit. The advantages of having a fine grained adaptivity is that we can have better power and timing performances trade-offs. If a certain block is failing timing then the voltage of that level will only be increased to meet the timing requirement of the circuit not for the whole circuit. In this way the circuit will have correct functional operation and will consume less power than the whole circuit working in elevated voltage levels. The advantage of having a fine grained adaptive dual supply voltage has been shown on a test chip with the fine grained adaptive circuit consuming 12% less power than a coarse grained configuration [18]. Along with having a better power-performance trade-off than a single supply voltage, the fine grained adaptivity techniques have also been shown to reduce the impact of process variations on the circuit with the standard deviation of delay being reduced from 8.1% to 6.1% [2]



(a) Coarse grained adaptivity



(b) Fine grained adaptivity

Figure 1.2: Coarse grained and fine grained adaptivity [18]

1.2 Motivation for the Study

With the recent trend of fine grained adaptivity we can have more control over the power and the timings of a circuit. But one of the major issue is the timing support for these type of circuits. A fine grained adaptive circuit can operate in different configurations. Let's say there are n number of blocks in a circuit and q number of different voltage levels in a circuit then we have q^n different adaptive scenarios in which the circuit can operate as we can see in the Table 1.1 where we have two blocks and two adaptive levels V_{low} and V_{high}.

Table 1.1: Different adaptive scenarios for 2 blocks and 2 adaptive levels

Block 1	Block 2
V_{low}	V_{low}
V_{low}	V_{high}
V_{high}	V_{low}
V_{high}	V_{high}

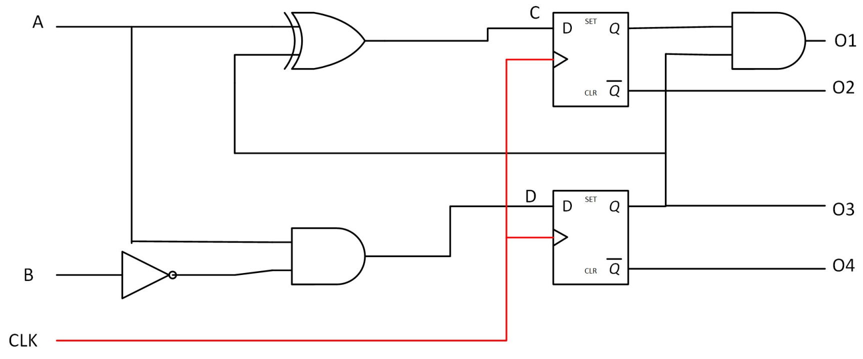
Clearly we can see that the problem is exponential in nature. For a particular scenario we can deduce the timing of that circuit using any of the SSTA techniques, which have been extensively studied [6, 9, 28]. If we want to know the timing yield of a chip with fine grained adaptivity no such work exist as per our knowledge. To calculate the timing yield we need to evaluate all the possible scenarios in which the circuit can work. Since the number of cases are exponential it can become impractical to deduce all the possible cases in a reasonable amount of time. Our study is focused on this problem. We want to evaluate the timing yield of a circuit with process variations and having fine-grained adaptivity in a reasonable amount of time. We explore different types of techniques to calculate the timing yield of the circuit.

2. TIMING ANALYSIS AND PREVIOUS WORKS

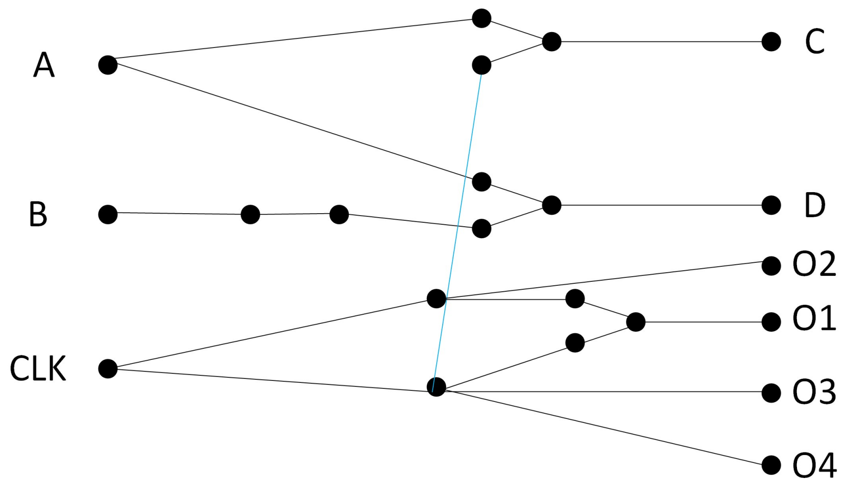
Timing analysis of a circuit is the method of analysing and validating the timing performance under given timing constraints. The analysis can be performed in either static or the dynamic way. Both form of timing analysis have a common goal of verifying whether a circuit functionally works correct under the given the timing constraint but the approach towards analysis is different. In dynamic timing analysis the circuit is simulated with actual inputs and verified that the circuit functionally works correct and has no timing violations i.e. all its paths should be stable when the data has to be sampled by flops at the next clock edge. The problem with this approach is that it is very time consuming and it is almost impossible to apply all the input test vectors. If the number of inputs are very large, which is usually the case in modern day designs, the number of input vectors become astronomical in number as the number of test vectors are exponential in terms of inputs.

Static timing analysis on the other hand uses the delay based calculations. This type of analysis uses the gate delays and compute the worst case timing paths for the circuit. It can be divided into two broad categories. One is path-based approach and the other is block-based approach. In path-based approach all the possible paths from the inputs to the outputs are evaluated. The number of paths can grow exponentially with the growing size of the circuit. Therefore this type of analysis is hardly used in practice.

The second type of static timing analysis approach is the block-based approach. In this approach, rather considering all the paths from the input to the output of a circuit we propagate the delay from the primary inputs towards the primary outputs using a PERT like traversal. This is usually done by using two mathematical operations - max and sum operations. At the input of a gate the max-delay of all the inputs is calculated using the max-operation. After this the result of the max-operation is added to the gate delay using the sum operation.



(a) A circuit



(b) Its timing graph

Figure 2.1: A circuit and its timing graph

For evaluating the circuit using the block-based approach we transfer the circuit into a timing graph as shown in Figure 2.1. To evaluate the timing of the graph we can use any of the established algorithm such as PERT. During evaluation to determine whether a circuit meets timing two types of violations in the whole circuit are checked which are set-up and hold time violations. Many different parameters are computed such as arrival time, required arrival time while propagating the delay from the input to the output. These are important to check for the timing violations and the to know which paths in the graph are failing.

Set-Up Time The minimum amount of time the data signal should be held steady before the clock event so that the data are reliably sampled by the clock. This applies to synchronous input signals to the flip-flop.

Hold Time The minimum amount of time the data signal should be held steady after the clock event so that the data are reliably sampled. This applies to synchronous input signals to the flip flop.

Critical Path The path between an input and output with the maximum delay.

Arrival Time The time elapsed from the primary inputs to a certain node in the timing graph.

Required Arrival Time The time when the data is required to be present at a certain node in the timing graph.

Slack The difference between the **Require Arrival Time** and **Arrival Time**

2.1 Delay Models

The delay of a gate is due to the electrical parameters of a gate which are the resistance and the capacitance of the gate. Different delay models can be used based on these parameters. One simple model is the Elmore delay model which is also used to model the interconnect delay. There are other complex and more accurate delay models but this model still provides a reasonable approximate delay at a very less computation cost than other models.

One other factor which affect the delay of the gate is the input slew rate. The slew rate is defined as the maximum rate of change of the voltage. If the input delay of the gate is large then the output of gate will also change slowly leading to higher delays than a input signal having lesser slew rate.

2.2 Statistical Timing Analysis

The effect of the process variations are becoming dominant in the modern circuits as the transistor dimensions are shrinking. The variation in dimensions of transistor i.e. length and width, doping concentration, gate oxide thickness etc. have led to variations in the delay of the gates. As a result the static timing analysis cannot be directly used in evaluating the circuit. One way is to use the worst case scenario for the delay under these variations with the static timing analysis. However if the worst case time of a gate is used for circuits with process variations then it has been found that the actual circuits are much faster than predicted by static timing analysis. This pessimistic approach has led to over design and higher costs.

Statistical timing analysis addresses this problem by using a statistical distribution for the delays rather than the worst case timing analysis. Generally Gaussian distribution is used to model the process variations which leads to a Normal distribution for the delay of the gate.

Statistical timing analysis have been an active research area. One of the earlier work which uses this approach is by Berkelar [5]. In this work he has proposed the use of Gaussian distribution for the delay of the gates instead of the triple best, typical and worst case delay. However, the author does not consider any correlation among the variations.

Many subsequent works have considered the issue of spatial correlation among variations. One work by Chang et. al. [8] which we have used for SSTA, consider the spatial correlation among different transistors using a grid approach. The delay distribution of transistors are modelled as Gaussian distributions and referred by the mean and the standard deviation. The PERT like traversal is used for propagating the delays. Two types of variations are considered in this work. One is the inter-die variations and the other one is the intra-die variations. The inter-die variations are the die-to-die variations and affect all the transistors on a chip in the same way. These type of variations are deterministic.

The intra-die variations affect the different transistors on a die differently. These variations can be divided into three components and can be modelled as a sum of are global, local and random components [8].

$$\delta_{intra} = \delta_{global} + \delta_{local} + \epsilon \tag{2.1}$$

The global component is dependent on the x, y co-ordinate location on the chip and is modelled as

$$\delta_{global}(x, y) = \delta_0 + \delta_x + \delta_y \tag{2.2}$$

The local component δ_{local} is proximity dependent and layout specific. It depends upon the proximity of a transistor to other transistors or other components on a chip such as capacitance etc. The random component ϵ is the random intra-chip variations and is modelled as a Gaussian distribution. Across the chip the random component has a correlated multivariate normal distribution due to the spatial correlation.

$$\vec{\epsilon} \sim N(0, \Sigma) \tag{2.3}$$

where Σ is the covariance matrix of parameters. The authors have only considered the global and the random variations, ignoring the local variations. Using this model the value of a parameter located at (x, y) has a delay as

$$p = \bar{p} + \delta_x x + \delta_y y + N(0, \sigma) \tag{2.4}$$

where $\bar{p}$ is the nominal design parameter value at die location $(0, 0)$. In this way, all parameter variations are modelled as location dependent normally distributed random variables.

The spatial correlations is modelled by using a grid model for the circuit as shown in Figure 2.2. The whole chip is partitioned into rectangular $nrow \times ncol$ grids. Inside a grid all the gates/transistors have perfect correlation. The gates lying in the neighbouring grids have a correlation < 1. Consider three grids A, B and C with the distance between grid A and C is greater than the distance between A and B then the correlation between A and C will be less than the correlation between A and B.

One other model to consider spatial correlation is the Quadtree model as in the work by [1]. The Quadtree model also partition the chip area into square grids. However the approach is to recursively partition the area into four equal parts. At each level each grid is again partitioned into four grids and the random variables of these grids have same correlation among them. The correlation between any two gates on the chip is the sum of all the random variables belonging to different grids.

However in this approach the correlation among different transistors can be different as in the Figure 2.3 the spatial correlation among the transistors in the grid 2.4 and 2.10 can be different from the correlation among the transistors in the grid 2.10 and 2.12 even if the spatial relation among these grids is the same.

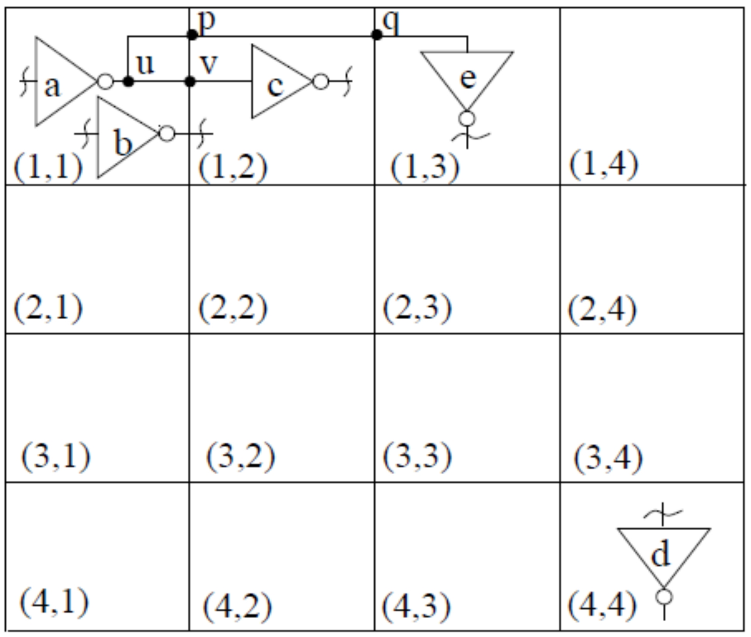


Figure 2.2: Grid model
[8]

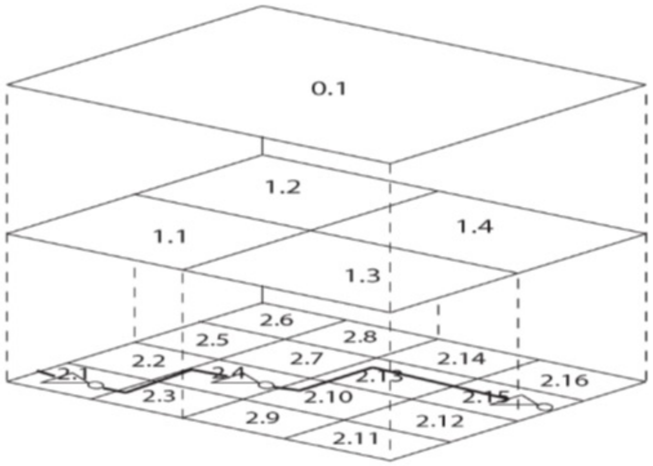


Figure 2.3: Quadtree model
[1]

The delay of a gate can be approximated linearly using the first order approximation. However the presence of spatial correlation can give rise to very complex equations as covariance between the variables has to be considered as we need to calculate the variance at the output of the gates. To solve this the authors in [8] have used principle component analysis to compute the principle components which are independent of each other and calculating the covariance between two delays becomes a simple mathematical operations in terms of principle components. Any delay value/variable can be expressed in terms of its principle components as follow.

$$d = d_0 + k_1 \times p_1^{'} + \cdots + k_m \times p_m^{'} \tag{2.5}$$

where the $p_i^{'}$ are the principal components of the various parameters.

The sum of two Gaussian random variables is Gaussian but the max operation's output is not Gaussian. The output of max function is approximated as Gaussian by using the Clark's approximation [10]. This gives results with small errors and a very good computation time. However the Clark's approximation can have very large error and in the work [25] have shown that the error can accumulate and can be large.

One way of SSTA is using the probabilistic events propagation approach as in the work by J. Liou et. al. [16]. In this work the delay variations are discretized into probabilistic events. Then these variations are added one by one to probabilistic delay events of a gate and grouped together to form the output of the gate as shown in Figure 2.4. These steps are carried out for whole the circuit till we get the events at the primary outputs. One main problem in this approach is the number of computations we have to perform for evaluating the circuit as the number of events grows with the increasing size of the circuit. The other problem is the number of computations required for evaluating re-convergent paths which are generally exponential in number. The authors have mentioned some techniques like ignoring the re-convergence paths which are very long as the re-convergent effect will be masked. However this approach does not provide much accurate results and could be slow for computation.

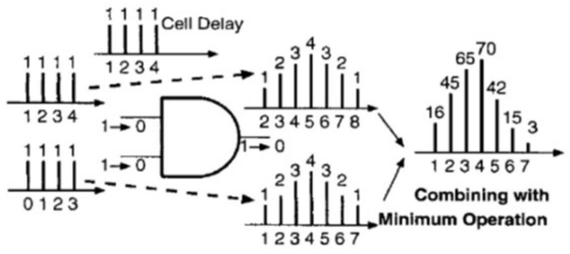


Figure 2.4: Probabilisic event approach
[16]

2.3 Hierarchical Timing Analysis

Hierarchical timing analysis is an important technique which helps in reduction of the computation time. In hierarchical timing analysis we extract the timings of a circuit with different input slew rates at the inputs. The number of timing paths from the input to the outputs are $no.\ of\ inputs \times no.\ of\ outputs$. The advantage of using hierarchical approach is that a circuit block or an IP can be used as black box when evaluated as a part of a larger circuit. This will reduce the computation time a lot if the same block is used many times in the larger circuit.

There has been research work in this area both in static and statistical approaches. One work by Bing Li et. al. [14] has shown that the timing models can be extracted using the statistical approach. The results show that the probability that a path will be critical or not, either tends to either zero or one. The authors have shown that the extracted model is $\sim 80\%$ smaller than the original circuit for the ISCAS'85 circuits.

The other work by C. W. Moon [17] has shown that the circuit can be reduced for the hierarchical timing analysis and the reduced circuit can be used to obtain the timing model of the circuit. Additionally the authors have proposed the use of gray-box instead of black-box. This way the number of paths can be reduced which are required for the hierarchical timing analysis. Since the size of the circuit is reduced this will aide in reducing the computation time.

The probability of a path becoming critical is given as in the work by Visweswariah et. al. [28] and also in [14]. The authors have used the tightness probability in the Clark's approximation for calculating the probability of a path becoming critical. This approach has led to a very important results. From the probability calculations we can determine which paths have higher probability of failing timing and which does not. Thus while evaluating the circuit for timing we can ignore the paths which will have almost zero probability of becoming critical.

The authors in [28] have also computed the required arrival time using the statistical methods and the probability of the paths for the required arrival time. This way one can calculate slack which will have a Gaussian distribution. This can be used in gate sizing and other optimization techniques.

2.4 Timing Yield

Given a timing constraint we have to determine whether a circuit meets the constraint or not. Across many chips the timing yield becomes the number of chips which will pass the timing constraints. Considering process variations the process parameters are no longer a fixed value but rather have a probabilistic distribution about the nominal value. Subsequently the gate and the interconnect delays become random values and also have a probabilistic distribution. Therefore, given a timing constraint the timing yield of a circuit is not a deterministic value but a statistical distribution. Therefore, it is natural to determine the timing yield of a circuit using the statistical methods.

Estimation of the timing yield of circuits with process variations have also gained attention. Many of the methods uses Monte Carlo approaches to estimate the timing yield of the circuit and many other uses analytical methods such as SSTA.

One such work by Min Pan et. al. [21] look at the problem of timing yield estimation for sequential circuits. They have proposed an algorithm which considers spatial and path re-convergence correlations of parameter variations, statistical longest and shortest path of the whole circuit and the clock skew caused by process variations to get the timing yield of the circuit. Their algorithm is based upon finding the longest and the shortest path of the circuit. However, they haven't mentioned or considered the fact that under process variations every circuit has a probability of becoming longest or shortest. The timing yield formulation for the sequential circuits as given by them is as follow.

$$\begin{aligned} Yield(T_{clk}) &= Prob(STM\ and\ HTM\ >\ 0) \\ &where \\ STM &= Set\ Up\ Time\ Margin \\ HTM &= Hold\ Time\ Margin \end{aligned} \tag{2.6}$$

Another work which evaluated timing yield by using Monte Carlo approach is by

Javid Jaffri [13]. This work explores the sampling based approach for calculating the timing yield of a circuit. They have shown advantage over the normal Monte Carlo based methods using the control-variate based technique with very few simulation iterations than crude or Quasi based Monte Carlo based methods or the order-statistics based estimator.

To estimate the timing yield for the FPGAs, some techniques have been proposed by Haile Yu et. al. [29]. This work focus on calculating the timing yield based on statistical timing analysis methods along with considering the spatial correlations among different logic blocks of an FPGA.

3. PROPOSED TECHNIQUES

3.1 Adaptive Circuit

An adaptive circuit is one which can change its supply voltage or body bias depending upon the time by which it is failing. The adaptive circuits can be designed and operated in either coarse or fine grained approach. However the fine-grained approach use is not is practice even after having benefits over coarse grained approach. One of the reason is the lack of timing support. Finding the timing yield of the circuit for all the adaptive scenarios will take lot of time given the large number of adaptive scenarios.

3.2 Problem Formulation

Given a combinational logic circuit design $\mathcal{C}$ composed by a set of adaptivity blocks $\{b_1, b_2, ..., b_n\}$, timing constraints $\mathcal{T}$ and certain delay models, a fundamental objective of timing verification is to find whether or not $\mathcal{C}$ satisfies $\mathcal{T}$. When variations are considered, the objective is often changed to find the probability that $\mathcal{C}$ satisfies $\mathcal{T}$, i.e., timing yield [6].

If $\mathcal{C}$ is an adaptive circuit, it includes variation sensors [11, 23], circuit tuning mechanisms [15, 26], and an adaptivity policy as shown in Figure 3.1. Without loss of generality, we assume the adaptive tuning is offline, i.e., it is performed at power-on or circuit idle time between normal operations. The sensors will detect the timing variations failure of a timing path and will generate an output vector which will be fed to the circuit which will be adaptive policy for the circuit. Thus the input to an adaptivity policy is an integer vector $\vec{x} = (x_1, x_2, ..., x_m)^T$ resulted from m variation sensors. For example, $x_i = 1$ means that sensor i detects a high risk of timing violation. The set of all possible sensor observation vectors is denoted by X.

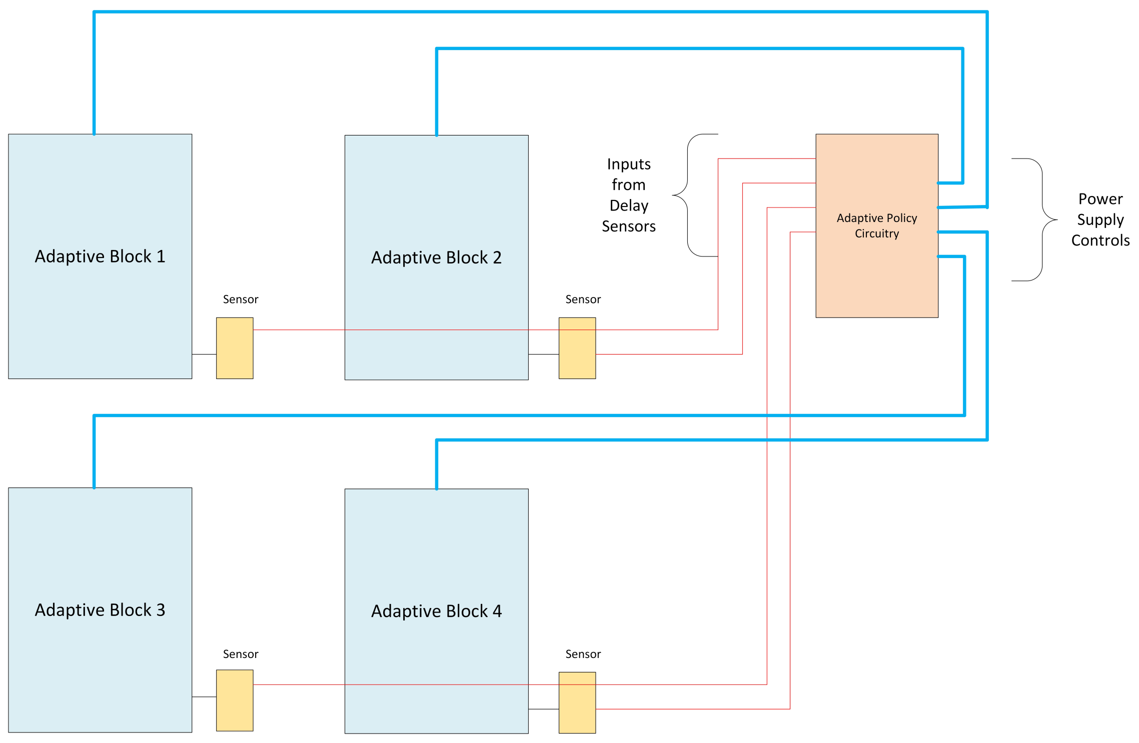


Figure 3.1: An adaptive circuit with adaptive policy

If there are n adaptivity blocks in $\mathcal{C}$, the output of adaptivity control is a vector $\vec{f} = (f_1, f_2, ..., f_n)^T$ where f_i specifies adaptivity configuration for block b_i. The value of f_i is an element of a set of adaptivity configurations $\{\phi_0, \phi_1, ..., \phi_q\}$, where ϕ_0 means no adaptivity action. For example, $f_j = \phi_0$ ($f_j = \phi_1$) chooses low (high) VDD for block b_j if there are only two adaptive levels available in the circuit. If a block is not able to run any adaptivity, its $q = 0$. All elements in the same adaptivity block follow the same adaptivity configuration. The total number of different adaptive configurations possible are therefore q^n which is exponential in the number of blocks. Lets represent the set of all possible adaptivity configurations is represented by F. Then the adaptivity policy can be described by a function $\Pi : X \rightarrow F$. This adaptive policy will tune to the circuit to a specific configuration so that none of the path is failing.

Timing verification for adaptive circuit is to find the probability that $\mathcal{C}$ satisfies $\mathcal{T}$ for a given adaptivity policy Π. This probability will the sum of all the probabilities of the circuit working in different possible adaptive configurations F.

We can apply different techniques such as hierarchical timing analysis for evaluating the timing yield of the circuit $\mathcal{C}$. Hierarchical Timing Analysis can provide us with a timing model for each block. However as discussed earlier hierarchical timing analysis has definite number of paths from the inputs to the outputs. But by considering variations, hierarchical timing analysis for adaptive circuit design is at least as complicated as statistical static timing analysis (SSTA) and the number of paths can become exponential as the critical paths are no longer deterministic as in conventional timing analysis. A non-critical path at the nominal case may become a critical path under variations. Such probability of a path becoming critical can be estimated by the method proposed in [28]. Now if every path has a certain probability of becoming critical then according to the path based analysis such an approach can be exponential in nature. Again the paths will have different critcality under different adaptive scenarios.

In a case where the number of paths are not exponential for the adaptive blocks as in [22], we still have to evaluate all the possible adaptive configuration for the circuit. This will reduce the time for evaluating a block but not the number of cases we might have to evaluate to calculate the timing yield of the circuit.

Besides the probabilistic variations, an adaptive circuit may autonomously change itself which will change all the timing values such as arrival time, required arrival time and slack. This fact makes the timing analysis for adaptive circuit even more difficult than SSTA. The delay values computed for one configuration will not be correct for another configuration due to which we might have to analyses the whole circuit again.

One another approach we can think of is using the probabilistic events approach [16]. In this probabilistic event approach we have to discretize the inputs and evaluate the whole circuit. The problem again here is that we need to evaluate all the possible adaptive configurations F. And again the added run time complexity of evaluating all the events which will keep on growing as we traverse the circuit.

One main observation from the above discussion is that we need to evaluate all the possible adaptive configurations F if we are not using the Monte Carlo approach. So our approach is based upon evaluating all the possible adaptive scenarios which we will discuss in the following sections.

3.3 Basic Approaches

When variations are considered, each component delay becomes a random variable. In this work, we assume the random variables follow Gaussian distribution, which is a reasonable approximation [6]. We consider spatial correlations among these variables using the grid model as given in [9].

A naïve approach is to use Monte Carlo simulation, where each run emulates one chip instance including its adaptivity actions and static timing analysis is performed for this instance.In order to obtain high statistical confidence level, the number of runs is typically very large. Gaussian random numbers are generated for each variation parameter's random varaible. The number of random variables for each parameter will be equal to the number of grids and the total number of samples for each random variable will be equal to the number of Monte Carlo runs we will be using for the simulations. Monte Carlo simulation will be used as a baseline for accuracy and runtime comparison in this work.

To emulate the adaptive actions in the Monte Carlo simulations an approach based upon the probability of each adaptivity block working in the different voltage levels can be used. Suppose an adaptive block has two voltage levels q_0, q_1 and the proabilities of working in these levels are P_0, P_1 respectively. Now in Monte Carlo simulations we can use the P_0, P_1 probabilities to assign an adptive action to a block.

Now we discuss an adaptivity scenario enumeration approach. In this approach, we run SSTA on the circuit for each adaptivity scenario individually by assigning different voltage levels to each block and then assemble the results into the overall timing yield. If there are m variation sensors and each sensor has up to p levels of output, there could be at most $|X| = p^m$ variation observation scenarios. If there are n adaptivity blocks and each block has up to q configuration options, there are at most $|F| = q^n$ configuration scenarios. Then, there are $\min(|X|, |F|)$ adaptivity scenarios.

SSTA is first conducted for the circuit without any adaptivity actions. Then, a probability density function (PDF) of timing slack can be obtained for each node of the circuit, including all sensor nodes. From these PDFs, one can estimate the probability of each sensor observation $P(\vec{x}_k)$, which is also the probability of an adaptivity scenario. If $|F| < |X|$, some adaptivity configuration must appear for multiple sensor observations according to pigeon-hole principle. Then the observations corresponding to the same configuration can be merged into a single adaptivity scenario.

By running SSTA on each scenario, we can obtain the yield $Y(\vec{f_k})$ for each configuration corresponding to observation $\vec{x}_k$. Then, the overall circuit timing yield can be estimated by

$$Y(\mathcal{C}) = \sum_{k=1}^{\min(|X|,|F|)} P(\vec{x}_k) \cdot Y(\vec{f_k}) \tag{3.1}$$

This SSTA-based enumeration technique is often faster than Monte Carlo simulation for small number of adaptive blocks. As the number of blocks increases and also the number of adaptive configurations q, the number of Monte Carlo runs needed will also increase as the number of samples at hand will increase a lot. However, it can still be very slow for fine-grained adaptivity, i.e., large n. If n is sufficiently large or the number of adaptive configurations q is also large, then due to the exponential nature of the number of adaptive scenarios, then the total number of runs required for enumerating all the possible scenarios can be larger than the number of runs required for Monte Carlo simulations. To solve the issue of large runs required for in the enumeration case and for further speed-up, we have used the circuit reduction, pruning and block merging techniques which we will discuss in the following sections.

3.4 Pruning and Reduction Techniques

The following techniques will help in reducing the number of enumerations for the different scenarios which will be required if we follow the SSTA based approach. These techniques are based upon some observations regarding the delay of a circuit. If a circuit is operating in higher voltage V_{high} then it will have lesser delay values as compared to the same circuit operating in a lower voltage V_{low} than V_{high}. This also shows that the if the circuit does not violate any timing constraint in V_{low} then it will also not violate the timing constraints in V_{high}.

3.4.1 Adaptivity Configuration Pruning

Suppose circuit $\mathcal{C}$ has n adaptivity blocks such that all gates in the same block follow the same adaptivity actions. We use $f(b_i)_j = \phi_j$ to represent adaptivity configuration j for block b_i. We define that $f(b_i)_j$ dominates $f(b_i)_k$ when $f(b_i)_j$ is a more powerful adaptive tuning than $f(b_i)_k$. For example, $f(b_i)_j$ uses higher supply voltage than $f(b_i)_k$, or $f(b_i)_j$ applies forward body bias (FBB) while $f(b_i)_k$ employs reversed body bias (RBB). We denote the dominance by $f(b_i)_k \prec f(b_i)_j$. If the case where $f(b_i)_j$ and $f(b_i)_k$ are identical is included, the notation becomes $f(b_i)_k \preceq f(b_i)_j$. Similarly, an adaptivity vector $\vec{f_j}$ dominates another one $\vec{f_k}$ if $f(b_i)_k \preceq f(b_i)_j, i = 1, 2, ..., n$. We use similar notation $\vec{f_k} \preceq \vec{f_j}$ for this definition.

The concept of dominance can be illustrated by an adaptivity graph as shown in Figure 3.2, where each row corresponds to an adaptivity block and each column indicates an adaptivity tuning effort level. Higher level implies a tuning with higher performance and more power dissipation. Then, each node indicates the tuning effort level for a block. An adaptivity configuration for entire circuit is a path that traverses one node on each row. For example, in Figure 3.2, the dashed red (thin) path chooses effort level 2 for block A, B and D, and effort level 3 for block C. One can tell the dashed red (thin) path is dominated by the solid green (thick) path.

Definition 1 – Robust adaptivity configuration: *An adaptivity configuration* $\vec{f_j}$ *for circuit* $\mathcal{C}$ *is robust if timing yield satisfies* $\sup Y(\mathcal{C}, \vec{f_j}) = 1$ *under this configuration.*

Observation 1: *If an adaptivity configuration* $\vec{f_j}$ *is robust, then any other con-*

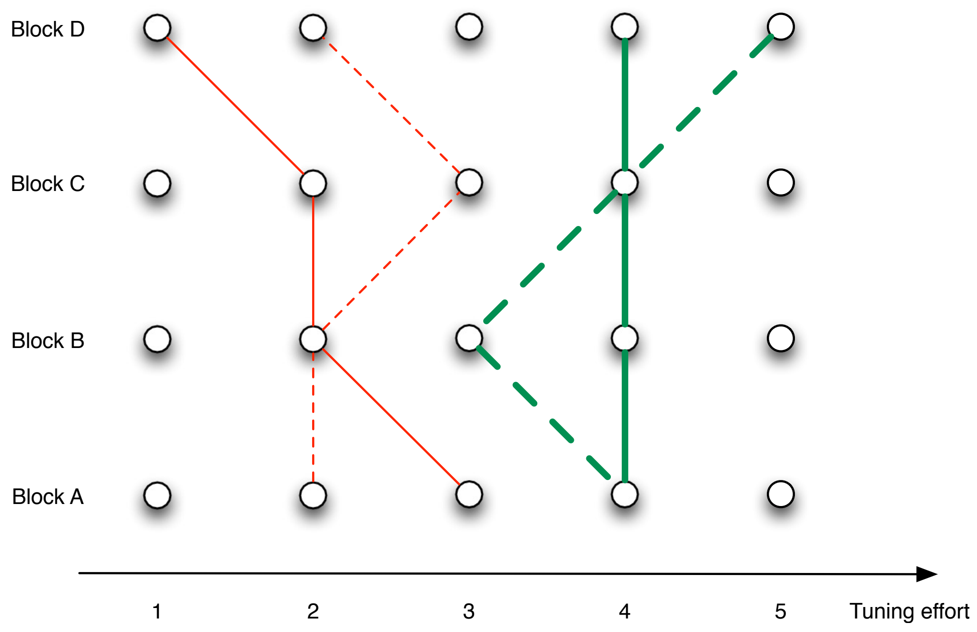


Figure 3.2: Adaptivity graph where each node tells the tuning effort level for an adaptivity block, and a path represents an adaptivity configuration for entire circuit.

figuration $\vec{f}_k$ satisfying $\vec{f}_j \preceq \vec{f}_k$ is also robust.

Definition 2 – Failing adaptivity configuration: *An adaptivity configuration $\vec{f}_j$ for circuit $\mathcal{C}$ is failing if timing yield satisfies $\inf Y(\mathcal{C}, \vec{f}_j) = 0$ under this configuration.*

Observation 2: *If an adaptivity configuration $\vec{f}_j$ is failing, then any other configuration $\vec{f}_k$ satisfying $\vec{f}_k \preceq \vec{f}_j$ is also failing.*

Definition 3 – Minimally robust adaptivity configuration: *An adaptivity configuration $\vec{f}_j$ for circuit $\mathcal{C}$ is minimally robust if $\vec{f}_j$ is robust and is no longer robust by any degradation of tuning effort level of any block.*

Definition 4 – Maximally failing adaptivity configuration: *An adaptivity configuration $\vec{f}_j$ for circuit $\mathcal{C}$ is maximally failing if $\vec{f}_j$ is failing and is no longer failing by any upgrade of tuning effort level of any block.*

In Figure 3.2, the two green (thick) paths are examples of minimally robust adaptivity configurations, and the two red (thin) paths illustrate maximally failing adaptivity configurations. Evidently, any configuration that dominates (is dominated by) a minimally robust (maximally failing) configuration can be pruned without examination. In Figure 3.2, any paths to the right (left) of any of the green (red) paths can be pruned.

3.4.2 Circuit Reduction

Circuit can be reduced from the timing point of view. For example some serial/parallel timing arcs can be merged [17]. Also, the timing arcs that are never critical even under variations can be neglected. A node in the circuit which has a positive slack under no adaptive action, will never become critical under any adaptive action. Since for timing yield we are concerned with the output nodes which have negative slack such nodes which have positive slacks starting from the output nodes can be pruned out in the circuit. By these reduction techniques, sometimes the timing graph of a circuit can be reduced by as much as 60%. The condition for removing a node from the timing graph is as follow

$$ReqAT = \mu_{ra} - 3\sigma_{ra} \tag{3.2}$$

$$ArrT = \mu_{arr} + 3\sigma_{arr} \tag{3.3}$$

$$prune = \begin{cases} 1 & ReqAT \geq ArrT \\ 0 & ReqAT < ArrT \end{cases} \tag{3.4}$$

For example in the Figure 3.3 if path A and C have negative slacks and paths B and D have positive slacks under no adaptive actions, then we can remove the paths

B and D from the timing graph as they will never have negative timing slacks under any adaptive action.

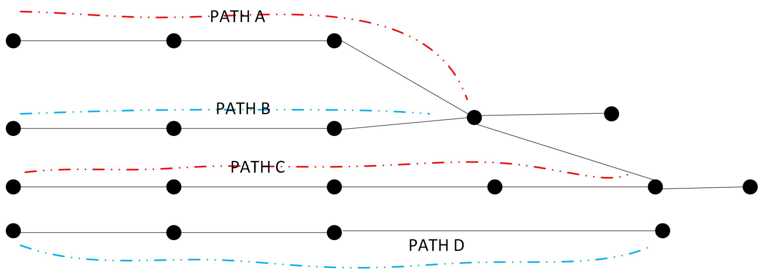


Figure 3.3: Circuit reduction techniques

Howeveer if we see that the paths A and B are the inputs to a Max function. If the mean values of these inputs are very close, then the mean of the output of the max function will be larger than either of these. But if the path B is removed then in subsequent SSTA of the circuit, there will be a small error introduced in the circuit. So while reducing the circuit we can add an small δ value to the condition while pruning the circuit as follow.

$$ReqAT = \mu_{ra} - 3\sigma_{ra} \tag{3.5}$$

$$ArrT = \mu_{arr} + 3\sigma_{arr} \tag{3.6}$$

$$prune = \begin{cases} 1 & ReqAT + \delta \geq ArrT \\ 0 & ReqAT < ArrT \end{cases} \tag{3.7}$$

3.4.3 Circuit Partitioning

The timing analysis complexity can be reduced if adaptivity blocks are independent of each other. This is illustrated by an extreme case in Figure 3.4, where the circuit is composed by n parallel and independent adaptivity blocks. In this case, the adaptivity configuration of a block does not need to be considered in conjunction with other blocks. If a block has q configurations, we only need to examine $q \cdot n$ scenarios in this example. By contrast, a full enumeration for a general case needs to check q^n scenarios.

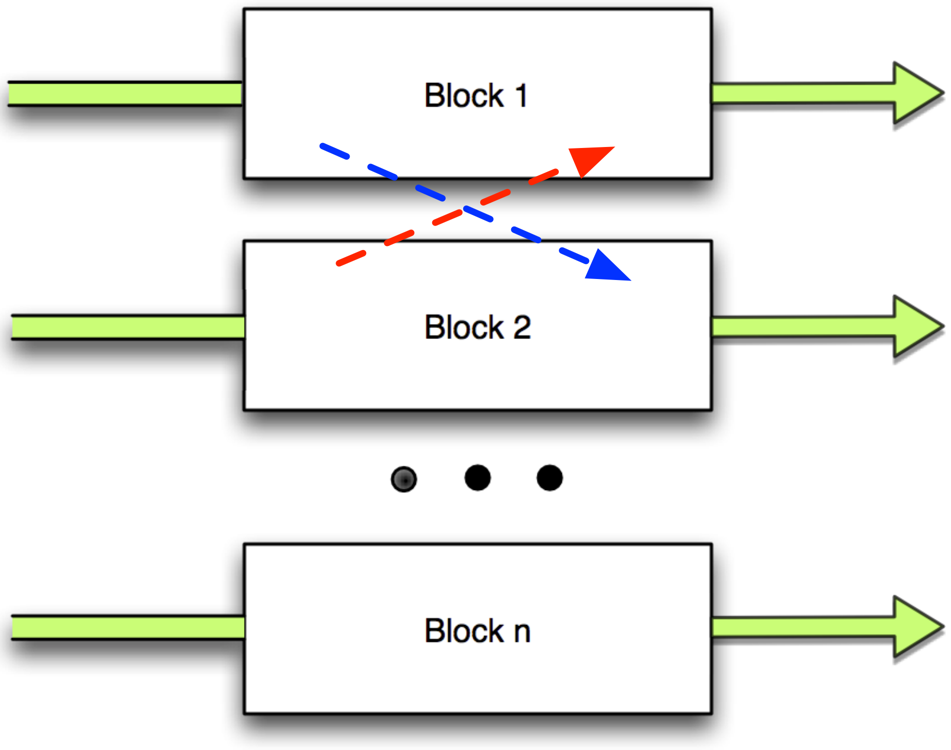


Figure 3.4: A circuit composed by n parallel and independent adaptivity blocks.

Of course, a case exactly the same as Figure 3.4 rarely happens in reality. However, partial separability should not be difficult to encounter. Even if there are signal paths connecting different blocks, indicated by dashed arrows in Figure 3.4, they can be reduced like discussed in Section 3.4.2, if they are never on timing critical paths.

3.4.4 Block Merging

Some blocks can be merged into a virtual block in the adaptivity enumeration without affecting accuracy. This is based on the following concepts and observation.

Definition 5 – Critical fan-in cone: *For a primary output O, its critical fan-in cone $\Gamma(O)$ is the set of blocks whose timing may affect the probability of satisfying timing constraint at O.*

Definition 6 – Mutually don't care: *For two primary outputs O_i and O_j, if block $b_k \in \Gamma(O_i), \notin \Gamma(O_j)$ and block $b_l \in \Gamma(O_j), \notin \Gamma(O_i)$, then b_k and b_l are mutually don't care blocks.*

Observation 3: *If two blocks are mutually don't care, they can be merged to a single virtual block during the adaptivity enumeration without affecting analysis results.*

We illustrate **Observation 3** by an example in Figure 3.5. Assume there is no path from pin u to pin v. Then, block C does not affect the timing at O_1 and block B does not affect the timing at O_2. Therefore, block B and C are mutually don't care. As such, we do not need to enumerate block B and C adaptivity separately. If there are two adaptivity options $\{0, 1\}$ for each block, then the enumeration like Table 3.1 covers 8 scenarios and all together is a complete cover for all relevant scenarios of O_1 and O_2, respectively. By the merging, the number of blocks is virtually reduced and therefore further speed-up can be obtained.

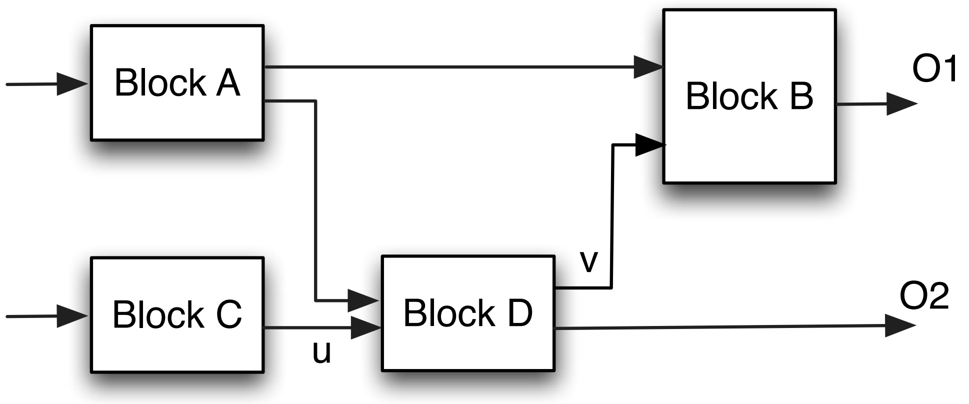


Figure 3.5: Block B and C are mutually don't care and can be merged into a single block during adaptivity enumeration.

Table 3.1: Adaptivity scenario enumeration for the example in Figure 3.5.

A	B & C	D
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

3.5 Overall Timing Analysis Algorithm

Our timing analysis algorithm starts from the exhaustive adaptivity enumeration described in Section 3.3 and is enhanced by the pruning and reduction techniques introduced in Section 3.4. The pseudo code of our timing analysis is shown in Algorithm 1. At the very beginning, we set the circuit to no adaptivity ϕ_0 and perform SSTA. We describe our algorithm with assumption that ϕ_0 is dominated by any other adaptivity configurations. This is not true for the case of reversed body bias, however, our algorithm can be easily extended to handle this case. After the initial SSTA, probability of sensor observations can be obtained. Then, according to the adaptivity policy, we can estimate the probability of each adaptivity configuration. In addition, the SSTA result can help the circuit reduction (step 3). Next, we partition the circuit according to Section 3.4.3. Steps 5-29 tell how to compute timing yield for each partition.

At the beginning for each partition, we first merge blocks according to Section 3.4.4. A set R_i is to keep track of all robust adaptivity configurations for this partition (step 7). Starting from all blocks with ϕ_0, we enumerate different adaptivity configurations with increasing number of changing blocks (step 8, 9 and 10). For example, if there are 4 blocks, we first examine configurations with only 1 block having adaptivity beyond ϕ_0 while all the other blocks remain at ϕ_0. There are $C_1^4 = 4$ such configurations. Next, we examine combination C_2^4 blocks with adaptivity beyond ϕ_0, and so on. For each combination of changing blocks, we enumerate all adaptivity configurations in a non-descending order of dominance. In steps 13 and 14, we apply a change while keep the other blocks at ϕ_0. If the new configuration dominates anyone in R_i or its probability is less than a threshold δ, we simply skip (step 16 and 17). Otherwise, SSTA is performed for this configuration. If this configuration is robust, it is added to R_i (step 21 and 22). At step 28, timing yield of a partition is obtained according to the yield of each configuration and probability of each configuration. Step 30 returns the final timing yield of the entire circuit.

```
Input  : Circuit $\mathcal{C}$ in $n$ blocks $B = \{b_1, b_2, ..., b_n\}$
         Adaptivity configuration options $\{\phi_0, \phi_1, ..., \phi_q\}$
         Adaptivity policy $\Pi$.
Output: Timing yield $Y(\mathcal{C})$
1  $f_i \leftarrow \phi_0, \forall b_i \in B$ ;
2  $SSTA(\mathcal{C})$ ; // Statistical static timing analysis
3  $\mathcal{C}^* \leftarrow Reduction(\mathcal{C})$ ; // Section 3.4.2
4  $\Psi = \{\psi_1, \psi_2, ...\} \leftarrow Partition(\mathcal{C}^*)$ ; //Section 3.4.3
5  for each $\psi_i \in \Psi$ do
6      $\psi_i^* \leftarrow Merge\_blocks(\psi_i)$ ; //Section 3.4.4
7      $R_i \leftarrow \emptyset$ ; // Set of robust adaptive configurations
8      for $j = 1$ to $|\psi_i^*|$ do
9          $S \leftarrow$ set of $C_j^{|\psi_i^*|}$ combinations of blocks;
10         for each combination $c \in S$ do
11             $F(c) \leftarrow$ all configurations of $c$ in non-descending order of
               dominance;
12             for for each $\vec{f} \in F(c)$ do
13                 Apply $\vec{f}$ to $c$;
14                 Apply $\phi_0$ to $\psi_i^*/c$;
15                 $f(\psi_i^*) \leftarrow$ current configuration of $\psi_i^*$;
16                 if ($f_r \preceq f(\psi_i^*), f_r \in R_i$) or ($P(f(\psi_i)) < \delta$) then
17                     Continue;
18                 end
19                 else
20                     SSTA($\psi_i^*$);
21                     if $\sup Y(\psi_i^*, f(\psi_i^*)) == 1$ then
22                         $R_i \leftarrow R_i \cup \{f(\psi_i^*)\}$;
23                     end
24                 end
25             end
26         end
27     end
28     $Y(\psi_i) \leftarrow \sum_{\forall f(\psi_i^*)} P(f(\psi_i^*)) \cdot Y(\psi_i^*, f(\psi_i^*))$;
29 end
30 Return $Y(\mathcal{C}) \leftarrow \prod_{\forall \psi_i \in \Psi} Y(\psi_i)$;
```

Algorithm 1: Timing analysis for adaptive circuit design.

4. EXPERIMENTS AND RESULTS

We evaluate the effectiveness of our approach by experimental comparisons on public benchmark circuits. Since there is no previous work on timing analysis for adaptive circuit, to the best of our knowledge, we compare the following approaches.

- Monte Carlo simulation. Spatial correlation among variations is considered. We use Monte Carlo simulation as a baseline for evaluating the accuracy and runtime of our approach.
- Exhaustive SSTA. SSTA is performed for every adaptivity configuration (Section 3.3).
- Ours. Adaptivity configurations are enumerated with pruning/reduction, and evaluated by SSTA (Algorithm 1).

Besides the circuit, voltage levels etc. an adaptive policy which decides the voltgae levels of different blocks is required as an input. Due to the lack of such a policy we have used simple calculations for calculating the probability of each block working in different voltage levels. The calculations of the probabilities is described in the next section.

4.1 Adaptivity Emulation Methodology

The input to our timing analysis is adaptive circuit design with certain adaptivity policy. However, there is no public benchmark of adaptive circuit. Therefore, we use conventional benchmark circuit and devise an adaptivity emulation methodology. In particular, this is to obtain a reasonable estimate of the probability of each adaptivity configuration. This methodology is based on statistical static timing analysis result on circuit without executing any adaptivity. We assume that the probability of a block configuration is independent of other blocks. This may not be exactly the case in practice, however, it is a reasonable approximation given that we do not have realistic adaptivity policy at hand.

Suppose the worst slack and delay in a block b are $\tilde{s}$ and $\tilde{d}$, respectively, in terms of mean and 3σ values. We only consider the case where $\tilde{s} < 0$ since the case of $\tilde{s} \geq 0$ does not need adaptivity. Then, the probability that the block goes to any configuration dominating ϕ_0 is estimated by

$$P(\phi_0 \prec f(b)) = \frac{|\tilde{s}|}{|\tilde{s}| + \tilde{d}} \tag{4.1}$$

Next, we will show how to estimate $P(f(b) = \phi_i)$ for each specific configuration level ϕ_i that dominates ϕ_0. This is a recursive procedure. If configuration ϕ_1 is applied, the slack is changed to $\tilde{s}_1$. If $\tilde{s}_1 > 0$, then $P(f(b) = \phi_1) = P(\phi_0 \prec f(b))$. Otherwise, we split $P(\phi_0 \prec f(b))$ into $P(f(b) = \phi_1)$ and $P(f(b) = \phi_2)$ if ϕ_2 is the maximum adaptivity level. The split is based on slack changes $\delta_1 = \tilde{s}_1 - \tilde{s}$ and $\delta_2 = \tilde{s}_2 - \tilde{s}$, due to the applications of adaptivity at levels ϕ_1 and ϕ_2, respectively. Then, the probabilities are given by:

$$\begin{aligned} P(f(b) = \phi_1) &= \frac{\delta_1}{\delta_2} P(\phi_0 \prec f(b)) \\ P(f(b) = \phi_2) &= \frac{\delta_2 - \delta_1}{\delta_2} P(\phi_0 \prec f(b)) \end{aligned} \tag{4.2}$$

If there are more adaptivity levels, we can divide the probability by using the same approach.

4.2 Experiments

The experiments are performed on ISCAS'85 and ISPD'13 [20] benchmark circuits, with the largest case having about $150K$ gates. We use the Elmore delay model and gate RC values are obtained from the ISPD'13 benchmarks. These circuits are placed by Feng Shui placer [3], where spatial correlation model is extracted. We consider process variations including gate length variation, whose standard deviation σ is 5% of nominal value, and gate width variation with σ being 2.7% of nominal value. Monte Carlo simulations are performed 10K and 20K iterations for small and large circuits, respectively. The probability threshold δ for neglecting an adaptivity configuration is 0.0001. More details on the adaptivity emulation methodology is provided in the Appendix. All of the methods are implemented in C++ and the program runs on a Linux server with 4 AMD Opteron processors of $2.2GHz$ operating frequency and $256GB$ memory.

The experimental results from ISCAS'85 circuits are shown in Table 4.1, 4.2, 4.3 and 4.4, with tight and loose timing constraints, respectively. In both cases, the relative error with respect to Monte Carlo from our method is about 0.9% compared with Monte Carlo results. Speedup of 32× and 64× are achieved for tight and loose timing constraints, respectively. Greater speedup is obtained for cases with loose constraints as they have less timing critical paths and allow more pruning and reduction. The speedup is not uniform among different circuits as the pruning/reduction techniques are highly dependent on circuit structure. Comparing with the exhaustive SSTA, our approach is several times faster and the result difference is one order of magnitude less than the error with respect to Monte Carlo method. This is because our method is derived from the exhaustive approach. The tables 4.1, 4.2, 4.3 and 4.4 also shows the number of gates in the circuit.

Table 4.1: Experimental results from ISCAS'85 circuits with tight timing constraints, comparison with respect to Monte Carlo

Circuit	#gates	Monte Carlo (MC)		Ours		W.r.t. MC	
		Yield	CPU (s)	Yield	CPU (s)	% Err	Speedup
c432	171	0.566	1.45	0.560	0.02	1.09	72.5
c499	218	0.586	1.86	0.590	0.14	0.68	13.3
c880	383	0.573	3.54	0.576	0.04	0.523	88.5
c1355	562	0.516	4.60	0.538	3.80	4.26	1.21
c1908	972	0.580	6.98	0.581	3.33	0.172	2.1
c2670	1287	0.563	10.56	0.568	0.51	0.88	20.71
c3540	1705	0.644	13.73	0.646	0.50	0.31	27.46
c5315	2351	0.597	21.30	0.598	0.32	0.16	66.56
c6288	2416	0.512	41.79	0.516	14.28	0.781	2.93

Table 4.2: Experimental results from ISCAS'85 circuits with tight timing constraints, comparison with respect to Exhaustive SSTA.

Circuit	#gates	Exhaustive SSTA		Ours		W.r.t. Exhaustive	
		Yield	CPU (s)	Yield	CPU (s)	% Err	Speedup
c432	171	0.560	0.02	0.560	0.02	-0.01	1
c499	218	0.590	0.18	0.590	0.14	-0.01	1.46
c880	383	0.567	0.20	0.576	0.04	-0.09	5
c1355	562	0.538	4.26	0.538	3.80	-0.04	1.12
c1908	972	0.587	10.54	0.581	3.33	-0.03	3.17
c2670	1287	0.568	1.16	0.568	0.51	-0.17	2.27
c3540	1705	0.646	1.63	0.646	0.50	0.04	3.26
c5315	2351	0.598	2.87	0.598	0.32	0	8.97
c6288	2416	0.516	15.01	0.516	14.28	-0.01	1.05

Table 4.3: Experimental results from ISCAS'85 circuits with loose timing constraints, comparison with respect to Monte Carlo

Circuit	#gates	Monte Carlo (MC)		Ours		W.r.t. MC	
		Yield	CPU (s)	Yield	CPU (s)	% Err	Speedup
c432	171	0.946	1.40	0.943	0.01	-0.31	140
c499	218	0.939	1.83	0.941	0.13	0.21	14
c880	383	0.960	3.55	0.953	0.03	-0.73	118
c1355	562	0.958	4.56	0.951	2.97	-0.73	1.5
c1908	972	0.929	6.87	0.924	2.51	-0.53	2.7
c2670	1287	0.918	10.67	0.917	0.27	-0.10	40
c3540	1705	0.961	13.78	0.958	0.09	-0.31	153
c5315	2351	0.917	20.75	0.912	0.21	-0.54	99
c6288	2416	0.942	41.27	0.937	5.81	-0.53	7.1

Table 4.4: Experimental results from ISCAS'85 circuits with loose timing constraints, comparison with respect to Exhaustive SSTA

Circuit	#gates	Exhaustive SSTA		Ours		W.r.t. Exhaustive	
		Yield	CPU (s)	Yield	CPU (s)	% Err	Speedup
c432	171	0.943	0.02	0.943	0.01	-0.09	2
c499	218	0.942	0.19	0.941	0.13	0.12	1.5
c880	383	0.953	0.20	0.953	0.03	-0.02	6.7
c1355	562	0.952	4.34	0.951	2.97	0.01	1.5
c1908	972	0.924	10.68	0.924	2.51	0.02	4.3
c2670	1287	0.918	1.17	0.917	0.27	0.03	4.3
c3540	1705	0.958	1.65	0.958	0.09	0.05	18
c5315	2351	0.912	2.89	0.912	0.21	-0.01	14
c6288	2416	0.938	7.40	0.937	5.81	0.03	1.3

Results from ISPD'13 circuits are displayed in Table 4.5, 4.6, 4.7 and 4.8. Again, the two tables are for different timing constraints. The speedup compared to Monte Carlo is hundreds and sometimes thousands. The error is greater, but still less than 4% most of time. Although the exhaustive SSTA approach is usually faster than Monte Carlo, its runtime scales poorly with circuit size and it cannot finish in three hours for a large case.

Table 4.5: Experimental results from ISPD'13 circuits with tight timing constraints, comparison with respect to Monte Carlo

Circuit	#gates	Monte Carlo (MC)		Ours		W.r.t. MC	
		Yield	CPU (s)	Yield	CPU (s)	% Err	Speedup
usb_phy	609	0.515	1.66	0.519	0.01	0.77	166
fft	32281	0.527	469	0.538	1.39	2.08	338
cordic	41601	0.582	476	0.573	1.47	-1.54	324
des_perf	112644	0.519	3146	0.540	41.38	4.04	76.02
matrix_mult	155325	0.576	4758	0.591	280	2.60	16.98
edit_dist	130661	0.515	3480	0.531	2.51	3.1	1386

Table 4.6: Experimental results from ISPD'13 circuits with tight timing constraints, comparison with respect to Exhaustive SSTA

Circuit	#gates	Exhaustive SSTA		Ours		W.r.t. Exhaustive	
		Yield	CPU (s)	Yield	CPU (s)	% Err	Speedup
usb_phy	609	0.518	0.08	0.519	0.01	0.20	8.0
fft	32281	0.536	19.05	0.538	1.39	0.37	13.79
cordic	41601	0.571	54.05	0.573	1.47	0.35	36.76
des_perf	112644	_	>3Hrs	0.540	41.38	_	_
matrix_mult	155325	0.589	1938	0.591	280	0.34	6.92
edit_dist	130661	0.531	1243	0.531	2.51	0	495

Table 4.7: Experimental results from ISPD'13 circuits with loose timing constraints, comparison with respect to Monte Carlo

Circuit	#gates	Monte Carlo (MC)		Ours		W.r.t. MC	
		Yield	CPU (s)	Yield	CPU (s)	% Err	Speedup
usb_phy	609	0.926	1.62	0.927	0.01	0.10	162
fft	32281	0.905	418.4	0.919	0.12	1.54	3486
cordic	41601	0.939	465.6	0.924	0.809	-1.59	575
des_perf	112644	0.900	2807	0.924	24.78	2.66	113
matrix_mult	155325	0.937	4053	0.963	8.39	2.77	483
edit_dist	130661	0.902	3572	0.938	0.62	3.99	5761

Table 4.8: Experimental results from ISPD'13 circuits with loose timing constraints, comparison with respect to Exhaustive SSTA

Circuit	#gates	Exhaustive SSTA		Ours		W.r.t. Exhaustive	
		Yield	CPU (s)	Yield	CPU (s)	% Err	Speedup
usb_phy	609	0.927	1.0	0.927	0.01	0	100
fft	32281	0.917	9.3	0.919	0.12	0.22	77.5
cordic	41601	0.924	52.26	0.924	0.809	0	64.6
des_perf	112644	_	>3Hrs	0.924	24.78	_	_
matrix_mult	155325	0.961	981.2	0.963	8.39	0.20	120
edit_dist	130661	0.937	1220	0.938	0.62	0.10	1968

We also compared Monte Carlo simulation and SSTA on mean and standard deviation (σ) of circuit delay for ISCAS'85 benchmark. The results are summarized in Table 4.9. One can see that the error of SSTA on the mean delay is almost always negligible. However, the error of SSTA standard deviation is quite significant. This is mostly due to the intrinsic error in the SSTA algorithm [9]. Even the raw variations follow Gaussian distributions, the timing variations after min-max operations are no longer Gaussian. The SSTA [9] uses Clark's method to approximate the non-Gaussian min-max delay by Gaussian distributions. This approximation brings errors and it always over-estimates the standard deviation. Therefore, a considerable part

Table 4.9: Comparison on mean and standard deviation of circuit delay.

Circuit	Monte Carlo		SSTA			
	Mean	σ	Mean	Error	σ	Error
C432	648	25.9	649	0.1%	26.5	2.3%
C499	329	12.6	326	0.8%	13.4	6.4%
C880	553	20.7	552	-0.03%	23.3	12.9%
C1355	572	21.1	573	0.09%	21.4	1.5%
C1908	744	25.5	744	0.05%	28.7	12.5%
C2670	624	18.8	624	0.04%	20.2	7.5%
C3540	824	28.5	824	0.02%	29.6	4.1%
C5315	756	24.5	757	0.2%	25.0	2.2%
C6288	1394	49.3	1396	0.1%	50.0	1.5%
Average				0.04%		5.7%

of errors from our method is attributed to the SSTA problem. The accuracy of our method is expected to be improved if more advanced SSTA techniques are employed.

5. CONCLUSION

Adaptive circuit design is a promising approach to handling variations with high power-efficiency. This work proposes the first systematic timing verification method to assist adaptive design, to the best of our knowledge. Since an adaptive circuit may have many configurations, analyzing all cases can be very time consuming. We propose a set of pruning and reduction techniques. These techniques are validated on benchmark circuits of up to $150K$ gates. The results show that our method provides order of magnitude speedup compared to Monte Carlo with very small errors.

REFERENCES

[1] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 900–907, 2003.

[2] K. Agarwal and K. Nowka. Dynamic power management by combination of dual static supply voltages. In *Proceedings of the 8th International Symposium on Quality Electronic Design*, pages 85–92, March 2007.

[3] A. R. Agnihotri, S. Ono, and P. H. Madden. Recursive bisection placement: Feng Shui 5.0 implementation details. In *Proceedings of the ACM International Symposium on Physical Design*, pages 230–232, 2005.

[4] V. E. Axelard and J. K. Kibarian. Statistcial aspects of modern ic design. In *Proceedings of the 28th European Solid-State Device Research Conference*, pages 309–321, 1998.

[5] M. Berkelar. Statistical delay calculation, a linear time method. *(Personal Communications)*.

[6] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. Statistical timing analysis: from basic principles to state of the art. *IEEE Transactions on Computer-Aided Design*, 27(4):589–607, April 2008.

[7] D. Boning and S. Nassif. *Design of High-Performance Microprocessor Circuits*. Wiley-IEEE Press, 2000.

[8] H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 621–625, 2003.

[9] H. Chang and S. S. Sapatnekar. Statistical timing analysis under spatial correlations. *IEEE Transactions on Computer-Aided Design*, 24(9):1467–1482, September 2005.

[10] C. E. Clark. The greatest of a finite set of random variables. *Operations Research*, 6(2):145–162, March 1961.

[11] A. Drake, R. Senger, H. Deogun, G. Carpenter, S. Ghiasi, T. Nguyen, N. James, M. Floyd, and V. Pokala. A distributed critical-path timing monitor for a 65nm high-performance microprocessor. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 398–399, 2007.

[12] D. Ernst, S. Das, N. S. Kim, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flaunter, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture*, pages 7–18, December 2003.

[13] J. Jaffari and M. Anis. Practical monte-carlo based timing yield estimation of digital circuits. *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 807–812, 2010.

[14] B. Li, N. Chen, M. Schmidt, W. Schneider, and U. Schlichtmann. On hierarchical statistical static timind analysis. *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1320–1325, 2009.

[15] X. Liang, G. Y. Wei, and D. Brooks. Revival: a variation-tolerant architecture using voltage interpolation and variable latency. *IEEE Micro*, 29(1):127–138, January 2009.

[16] J. J. Liou, K. T. Cheng, S. Kundu, and A. Krstic. Fast statistical timing analysis by probabilistic event propagation. In *Proceedings of the Design Automation*

Conference, pages 661–666, 2001.

[17] C. W. Moon, H. Kriplani, and K. P. Belkhale. Timing model extraction of hierarchical blocks by graph reduction. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 152–157, 2002.

[18] A. Muramatsu, T. Yasufuku, M. Nomura, M. Takamiya, H. Shinohara, and T. Sakurai. 12% power reduction by within-functional-block fine-grained adaptive dual suuply voltage control in logic circuits with 42 voltage domains. In *Proceedings of the ESSCIRC*, pages 191–194, 2011.

[19] S. R. Nassif. Design for variability in dsm technologies. In *Proceedings of the IEEE International Symposium on Quality Electronic Design*, pages 451–454, March 2000.

[20] M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, and C. Zhuo. An improved benchmark suite for the ISPD-2013 discrete cell sizing contest. In *Proceedings of the ACM International Symposium on Physical Design*, pages 168–170, 2013.

[21] M. Pan, C. C. Chu, and H. Zhou. Timing yield estimation using statistical timing analysis. *Proceedings of the Midwest Symposium on Circuits and Systems*, 2005.

[22] A. Ramalingam, A. K. Singh, S. R. Nassif, G. Nam, M. Orshansky, and D. Z. Pan. An accurate sparse matrix based frameork for statistical static timing analysis. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 231–236, November 2006.

[23] T. Sato and Y. Kunitake. A simple flip-flop circuit for typical-case designs for DFM. In *Proceedings of the IEEE International Symposium on Quality Electronic Design*, pages 539–544, 2007.

[24] K. N. Shim and J. Hu. Boostable repeater design for variation resilience in VLSI interconnects. *IEEE Transactions on VLSI Systems*, 21(9):1619–1631, September 2013.

[25] D. Sinha, H. Zhou, and N. V. Shenoy. Advances in computation of the maximum of a set of gaussian random variables. *IEEE Transactions on Computer-Aided design of integrated circuits and systems*, 26(8), August 2007.

[26] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. *IEEE Journal of Solid-State Circuits*, 37(11):1396–1402, November 2002.

[27] J. W. Tschanz, S. Narendra, R. Nair, and V. De. Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performannce microprocessors. *IEEE Journal of Solid-State Circuits*, 38(5), May 2003.

[28] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, D. K. Beece, J. Piaget, N. Venkateswaran, and J. G. Hemmett. First-order incremental block-based statistical timing analysis. *IEEE Transactions on Computer-Aided Design*, 25(10):2170–2180, October 2006.

[29] H. Yu, Q. Xu, and P. H. W. Leong. On timing yield improvement for fpga designs using architectural symmetry. *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pages 278–278, 2011.