Learning based SSTA

Zhiyu Chen
zhiyuc@usc.edu

Hsu Cheng
hsucheng@usc.edu

Harmanpreet Singh Kalsi
hkalsi@usc.edu

***Abstract*—Time closure issue has always been an essential part of VLSI design process. Static Timing Analysis(STA), though widely adopted as the main stream of computing expected timing behavior of digital circuits, is now facing increasing challenges when applied on advanced process nodes. Among all the challenges, the incresaing uncertainties in process variation both for the font-end and back-end design has significantly increased the complexity of performing STA, also, weakened the accuracy of predicted delay model. Therefore, to get accurate delay mode for most concurrent process nodes, the Statistical Static Timing Analysis(SSTA), has been proposed. Compare with STA, which treats the delay as a certain value, the SSTA modeled delay as distributions and find following delays through distribution calculation. Due to its ununified nature, SSTA, provides more realistic and accurate results when dealing with circuits with more pronounced process variation.**

I. Introduction

The SSTA, first proposed more than a decades ago, has been considered as a potential solution for dealing with the growing issue of process variation in advanced process nodes. In the SSTA setting, all delays are modeled in distribution format, where the latest arrival-time distribution at the sink node can be found by propagating the arrival time from the source node through the timing edges while computing the latest arrival-time at every node in topological order. Subsequently, the latest arrival-time disrtibution at the sink node is the circuit-delay distribution. [1] Since, all delay are represented in distribution model, unlike STA, no pessimistic assumptions need to be made in SSTA. In addition, there is no need for considering all the increasing amount of corner cases counting for all correlations between physic parameters. Thus, provides a more realistic and accurate delay estimation for the overall circuit.

Currently, there are three different models in acquiring the statistical distribution data for SSTA, they are Statistical Corner Model(SCM), TCAD-based Statiscial Model(TCAD) and Monte Carlo Model(MCM). In SCM, data is extracted from a vest amount of test runs of target technology within different dies, wafers, temperature and many other parameters. This data then forms the delay distribution of the specific device. While in TCAD, very similar to the SCM model, it generates the initial data through simulations of CAD tool and refine the data after real production. The MCM model, however, makes the assumption that all device parameters are independent. Based on this hypothesis, it runs a series of simulation runs where device parameters are randomly generated based on the distribution of those parameters in model. [2]

Similar to modeling methods, there are mainly two categories of SSTA implementations, Path-based and Block-based SSTA. For Path-based SSTA, a certain set of path with the largest possibilities of being critical path, would first be selected from the entire circuit. Then delay computation would only be performed on these selected path. While for the Block-based SSTA, it propogates through the whole circuit based on a topological traversal. Then, delay for each sink point in the circuit will be calculated. In this article, Block-based method is implemented.

Despite all the improvements provided by SSTA, there are still issues stalling SSTA from taking the place of traditional STA for doing timing analysis. One of the most pronounced issue is on how to effectively deal with both the topological and spatial correlations while performing SSTA analysis. Other concerns including the vast amount of computational effort need to pay for doing the distribution computation in getting the final delay model.

This report will be separated into three part, the first part, including section I, II, III, and IV, will introduce the concept of SSTA and will discuss some backgrounds for performing SSTA analysis. This part will also contain details about the implementation and corresponding result of proposed software simulator for SSTA analysis. The second part which is section V, Literature Review, presents some most recent ideas gathered from other articles that might be helpful on improving current design. While the last part including the rest of the sections, list plans for future phases as well as work distribution among the team members.

II. Background

In SSTA, all delays are modeled in probability distributions. For finding a delay in a certain point of the circuit, operations such as summation and maximization need to be done. In short, the delay at a given node of the circuit should equal to the maximum of sum of all the previous paths delay plus the delay of the node. Or it could be the maximum of all inputs delay added to the delay of this node. Therefore, Sum and Max operations of distributions is need for SSTA calculation. In this section, mathematical equations for these two operations will be presented and explained. The circuit model applied in this project will be introduced in the part as well. In addition, both spatial and topological correlations will also be discussed.

A. Summation

To find the sum between two distributions, convolution need to be done, where Fig. 1, shows the definition of f convolve g where f and g are two random functions.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{1}$$

Here if f and g are treated as two distributions, it is saying that, the resulting distribution of the sum operation will have a probability of $(f{*}g)(t)$ at time stamp t, which equals to the sum of products of all the probabilities of the point in f and g where time stamps of f and g adds up to t.

Since for implementation, all distributions are treat as discrete distributions. Assuming X and Y are two discrete distributions, and Z is the sum of X and Y, the relationship between X, Y and Z can be represent as Eqn. 2.

$$Z(t) = \sum_{t=0}^{\infty} X(\tau)Y(t-\tau) \tag{2}$$

B. Maximization

To find the maximum between two distribution models, let's first assume two random variables X and Y. Then if Z is defined to be the maximum between X and Y we could write the equation as:

$$Z = max(X, Y) \tag{3}$$

Which could further be separated into two cases:

$$X \geq Y \ then \ Z = X \ denote \ as: \ A \tag{4}$$

$$X \leq Y \ then \ Z = Y \ denote \ as: \ \bar{A} \tag{5}$$

Therefore, the distribution function (CDF) of Z at point t can be expressed as:

$$F_z(t) \ = \ P(Z \leq t) = P(Z \leq t, AU\bar{A}) \tag{6}$$

Since A & $\bar{A}$, are two disjoint events, with some manipulations, we can get:

$$F_z(t) \ = P(X \leq t, X \geq Y) + P(Y \leq t, X{<}Y) \tag{7}$$

Then, if we look at this equation from a graphic point of view, Fig.1:

We can see that this equation, Eqn.7 also equals to:

$$F_z(t) = P(X \leq t, Y \leq t) \tag{8}$$

Knowing X and Y are two independent events, we get:

$$F_z(t) = F_{X,Y}(t, t) \tag{9}$$

Base on the fact that the density function (PDF) is the derivative of distribution function (CDF), if we do the derivation on both sides by t, we get the density function of Z equals to:

$$f_z(t) = d/dt(F_{X,Y}(t, t)) = f_x(t)F_y(t) + f_y(t)F_x(t) \tag{10}$$

If we take a further look at this function, Eqn.3, we find that it is saying the possibility of the maximum value at point t equal to the possibility of that point in X, times the total possibility of the value less than t in Y. Plus the possibility of that point in Y, times the total possibility of the value less than t in X.

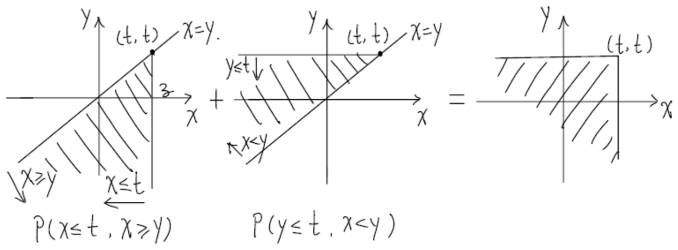


Fig. 1. Eqn.1 from a graphic point of view

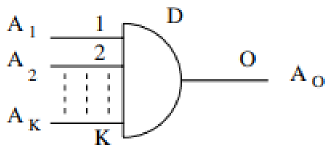


Fig. 2. Circuit diagram for AND gate with k inputs [3]

C. MAX of SUM & SUM of MAX operation

To find if there are difference between these two operations, an experiment has been conducted in Phase I. Base on the results, for now it is conclude that these two operations provide very similar but different results for calculating the delay mode. For this implementation, the SUM of MAX operation is chosen due to the fact that Max operation takes less computational effort compare to Sum operation.

D. Circuit model

For calculating the delay model at each sink point in the circuit, the circuit is modeled base on each every gate elements. For every single gate, to get the delay distribution at the output node, the maximum delay among all the inputs will first be selected and will than be added with the self delay of the gate. Fig. 2 illustrates a circuit diagram of a k inputs AND gate. For every input, lets denote the delay is D_i, where i range from 1 to k. And for the output A_0, denote the delay as D_0. For the gate itself, denote the delay as D_G. Therefore, D_0 is given by Eqn.11. Noticing that this equation also equals to MAX(D_1+D_G,...,D_k+D_G), which has been proved in Phase I report.

For solving the topological correlation, topological sort is applied in the circuit model for leveling the circuit. Then, the delay while be computed base on the level order. Fig. 3 gives an example of leveled circuit. Based on Fig. 3, G1 and G2 will first be calculated as they are at the lowest level among all gates. Then G3 and G4 will be processed and finally G5 and G6. By doing this, it ensure the input delay for each gate is already computed when processing it.

$$D_0 = max(D_1, ..., D_k) + D_G \tag{11}$$

E. Spatial correlation

Spatial correlations, usually refer to the within-die variation imposed during the manufacture process, are parameters to

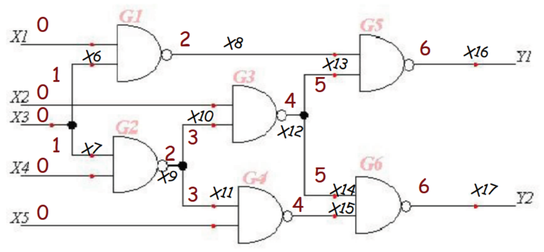


Fig. 3. Example of leveled circuit

represent the relative dependencies of delay distributions of individual circuit components due to their locations on the circuit board. [4]. Typical within-die variations may results from lenses imperfections, mask perturbations and variations of the laser intensity. For instance, a perturbation in the width of a wire will affect both the resistance and capacitance, where a decrease in resistance usually coupled with an increase in resistance. As these two electrical attributes both have effect on the delay distribution, simply ignoring this correlation will result in an inaccurate result.

Therefore, to better dealing with these correlated parameters, the spatial correlation parameter is introduced to represent the spatially correlated within-die variations. With this in mind, the delay distribution along one single path with n number of stages of equal delay distribution (same mean μ, same standard deviation σ^2) can be represented using, eqn.12 and 13, where ρ, represents spatial correlation factor. [5]

$$\mu_{path} = n\mu \tag{12}$$

$$\sigma_{path}^2 = \sum_{i=1}^{n} \sigma^2 + 2\rho \sum_{i=1}^{n} \sum_{j>i}^{n} \sigma_i \sigma_j = n\sigma^2(1 + \rho(n-1)) \tag{13}$$

By observing eqn.12 and 13, one can see that the mean value of the delay does not affected by the correlation factor, while the standard deviation increased with the increasing correlation. Meanwhile, if ρ = 0, then the two distributions are independent, where the σ_{path} will just be nσ^2. What's more, Arthur of [4] also proved that the correlation between random variables also impacts the skew of the resulting distribution after MAX operation. In short, amount of skew decreases with the increasing correlation.

Base on the effect of correlation factor on MAX operation, if timing analysis is done by ignoring the correlations between parameters (assuming all random variables are independent), then the total delay at output will be a worst case result, with maximum skew.

Many previous studies have suggest multiple solutions in dealing with spatial correlations, where all of them share common concept of partitioning the die into sub-blocks, assigning the same random variables to all the components within the same partition and find the spatial correlations between each partitions. Among all the suggested method, the most trivial one is proposed in [6], where the die is partitioned based on a grid model. More advanced approaches also involves quadtree model [7], where the entire die is separated into layers of sub-parts each with four components. In this model, the die will be split down to gate level. where each sub-parts only contain one gate. Base on this quadtree model, the correlation between each partitions than decreases with the up rising of layers. Although relatively intuitive, both grid and quadtree models only considered linear cases. For better accommodating the situation when the delay information is nonlinear or non-normal, a Taylor-series expansion-based polynomial representation of gate delays and arrival times is discussed for effectively capturing the nonlinear dependences.

Based on the Dependence Propagation Approaches, in the Taylor-series model, the author uses P$_i$, Q$_i$, and R$_i$ to represent the mutually independent location-dependent parameters random variables that affects the gate delay. Therefore, the delay of gate i can be modeled as a function of these independent parameters as given by Eqn.14. Then the author proposed a spatial correlation model, which partition the gates into spatial regions, Fig. 4 and assigned each corner of the circuit graph as P$_i$. Thus, for any gate j, it can be moded by corresponding parameter P$_j$ as given by Eqn. 15 where a$_0$ is the nominal value of parameter P$_j$. The coefficients a$_1$, a$_2$, a$_3$, a$_4$ can be computed by using the radial distance of R$_1$, R$_2$, R$_3$ and R$_4$ shown in the Fig. 4 Hence, the gate delay can be modeled as a function of Eqn. 16. For ease of presentation, lets representing these variables as Y$_j$ where j range from 1 to 12. Using Taylor-series expansion about the mean values, the gate delay D$_i$ from Eqn. 14 can also be present as Eqn. 17 The nominal values for the gate delay happens when all Y$_i$ variables are zero (essentially no variance). Therefore D$_i$$(nominal)$ = G$_i$(0) [3]

$$D_i = F(P_i, Q_i, R_i) \tag{14}$$

$$P_j = a_1P_1 + a_2P_2 + A_3P_3 + a_4P_4 + a_0 \tag{15}$$

$$D_i = G_i(P1, P2, P3, P4, Q1, Q2, Q3, Q4, R1, R2, R3, R4) \tag{16}$$

$$D_i = G_i = G(0) + \sum_{k=1}^{12}(Y_k)G^{'}(0) + \frac{1}{2!}\sum_{k=1}^{12}(Y_k)^2G^{''}(0) + ... \tag{17}$$

F. Topological correlation

In addition to the spatial correlation, when performing SSTA analysis, one should also consider the topological correlation raising from the re-convergent path in the circuit. The two path marked in red, is an illustration for convergence in circuit. Here, since the two input to the g_3 gate have a common path before g_1 and g_2, one could not treat the accumulated delay at the two inputs of g_3 as independent. Therefore, Eqn.10 for finding maximum between these two inputs could not directly applied here. Or otherwise, this will result in a worst case

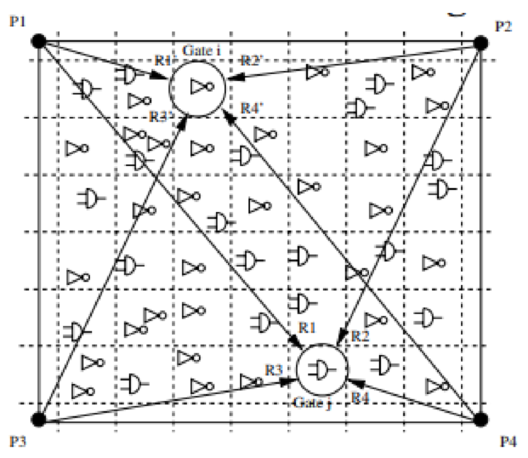


Fig. 4. Grid-Based Spatial Correlation Model [8]

delay with maximum skew same as described in the spatial correlation part.

To deal with this re-convergence issue, one way is to split the accumulated arrival delay information at the inputs of gate g_3 into two separate parts. The first part contains the arrival delay information upon the divergent node of the two path, which in Fig.5 is the node at the output of gate g_0. Besides, the second part contains the two separate path from the divergent node to the inputs of the convergent node, which in Fig.5, is the output of gate g_3. Since the delay information for the second part are independent to each other, Eqn.10 could be applied to find the maximum delay between the two path, which will then be added back to the total arrival delay at the divergent node using summation. To apply this concept of breaking arrival delay into two parts and perform maximum operation on the independent path, [9] suggested a Dependency List (DL) approach. The DL, as suggested, contains each vertex in the timing graph which lists the vertices on which the arrival time of the current vertex depends. The vertices are arranged in an descending order based on level, with the most recent vertex appears first. The propagation of the DL is based on a DLPropagate algorithm proposed by the aurthor as shown in Fig.6, where DL_i denotes the DL of the ith input and DL_o denotes the DL of the output node. Two heuristics also used here to limit the size of the DL. Users are allowed to specify the size of the list, and are allowed to only carry-forward the n most recent vertices. After that, to calculate the arrival time at the ourput of a re-convergent gate with multi-input each with possibly a non-empty DL, an approximate algorithm in Fig.7 is provided to find the top most vertices.

Other methods in dealing with topological correlations also proposed in previous studies. In [10], the authors uses the statistical sum operation to reduce series edges in the timing graph to a single edge, while keep recomputing the edges with nonzero correlation to the reduced edge. Similar reduction procedure also apply for parallel edges using statistical maximum operation while assuming normal distribution base on

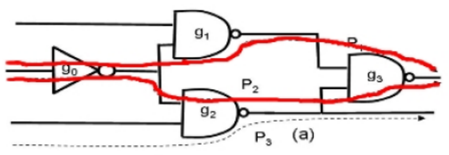
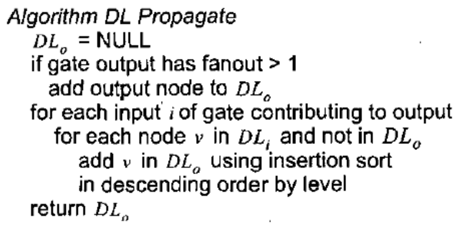


Fig. 5. Toplogical Correlation



```
Algorithm DL Propagate
  $DL_o$ = NULL
  if gate output has fanout > 1
    add output node to $DL_o$
  for each input $i$ of gate contributing to output
    for each node $v$ in $DL_i$ and not in $DL_o$
      add $v$ in $DL_o$ using insertion sort
      in descending order by level
  return $DL_o$
```

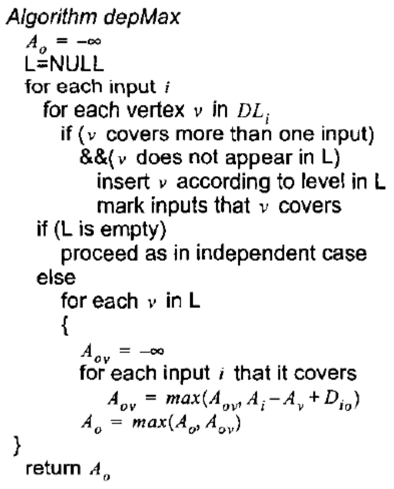
Fig. 6. Algorithm to propagate dependency list [9]

the results provide in [11]. This approach limits the correlation information of the edges that are stored in the memory by eliminating those that are no longer required. This idea is extended by [12] and [13] by legalizing the directed acyclic graph (DAG) like circuit using a depth first search. Where level is used to identify the nodes whose correlation information can be discarded at a given stage of the circuit.

In this implementation, method for dealing with topological correlation has been applied following the idea suggested in [9]. Both DLPropagate and the approximate algorithm from Fig.6 and 7 are applied in Python.

III. SOFTWARE IMPLEMENTATION

In the project, the Distribution propagation approach for Block-based SSTA has been implement in Python. While a paralled C++ program will transit the gate level Verilog file into bench file to serve as the input of the circuit information to



```
Algorithm depMax
  $A_o = -\infty$
  L=NULL
  for each input $i$
    for each vertex $v$ in $DL_i$
      if ($v$ covers more than one input)
        &&($v$ does not appear in L)
          insert $v$ according to level in L
          mark inputs that $v$ covers
  if (L is empty)
    proceed as in independent case
  else
    for each $v$ in L
    {
      $A_{ov} = -\infty$
      for each input $i$ that it covers
          $A_{ov} = max(A_{ov}, A_i - A_v + D_{io})$
      $A_o = max(A_o, A_{ov})$
  }
  return $A_o$
```

Fig. 7. Algorithm to compute output arrival time in presence of re-convergent path [9]

the platform. In this implementation, it is assumed delay distribution to be normally distributed and only topological correlation is taken into consideration. This section will present the implementation and discuss on possible improvements.

A. Overview of software implementation

Fig. 8, shows a top-level view of the overall software implementation. As demonstrate by this diagram, the overall implementation could be separated into two parts, the prepossessing part, including the C++ part that transit Verilog file into bench file and the circuit legalizing part, implemented by Jiaming Li and rest of the ATPG team. The overall prepossessing part reads the ciruict information through Verilog file and generate leveled circuit model as output. The SSTA calculation part, takes in both the generated model from prepossessing and the delay information for each gate from 'sstalib', a library stores the delay information, through a function called 'read_sstalib'. Then, based on both inputs, the PDF class will first generates a leveled ciruit model with delay information stored at each every circuit node. After that, this delay distribution will be propagate through out the circuit in the leveled order. And finally will generate the overall delay model for the circuit with taking topological correlation into consideration through the DL method described in the background section. Detailed explain for each part of the design will be discussed in the following part of this section.

B. Preprocessing

As aforementioned, the main function of this part is to take in the circuit information from the Verilog file, translate into bench file through C++, and then output the leveled circuit model in a list of circuit objects. Each object in this list represent an circuit element/ gate, which contains information such as gate type, input nodes, output nodes and level number. The reason for leveling the circuit is for helping solving the topological correlation issue. The method currently implemented by ATPG team to do the leveling is brute force. While could be improve with other topological sort algorithm such as Depth First Search (DFS) to improve the performance.

C. SSTA calculation

The concept of Object Oriented Programming has been utilized in this part implantation. Then core of this design is the PDF class, which contains constructor for generating PDF object for a certain node in circuit along with several helper functions for calculating the propagation of the delay distribution.

A typical flow for the SSTA calculation part can be described as follow. First, The PDF class will take in the list of circuit objects generated by the prepossessing part, as well as the delay information getting from the sstalib through the 'read_ssta' function. Then, based on the gate type and corresponding delay, the PDF class will generate a list of discrete PDF for each of the element in the list of circuit object, which is the 'Leveled Circuit model (With delay information)' shown in Fig. 8. After knowing all the delay distribution of particular gates in the circuit, the SUM and MAX function from the PDF class will then be used for calculating the propagated delay information along the circuit model in a level by level order while considering the topological correlation between inputs using the dependency list method. Finally, after traverse through the entire circuit, the entire SSTA model is constructed for the circuit.

Some of the most important functions such as NORM, SUM, MAX, data_shrink function as well as the implementation of DL algorithm will be discussed in detail below.

1) NORM: The NORM function has four inputs, it takes in the delay information of mean and standard deviation , as well as the sample distance and sample size. The delay information are taken from the 'sstalib' library while the sample distance and sample size, are parameters assigned by the operator in the 'config.py' file. Base on these four parameters, the NORM function generates a fixed resolution discrete delay distribution for each every inputs by evenly distribute the sampling points on both side of the mean value.

Worth noticing that, since samples are taken from the normal distribution. To ensure the total probability of all samples add up to 1, a re-normalization process has been done, base on Eqn. 18, where $X(t)$ represents the probability at time delay t and $X(t)'$ represents the probability at time delay $X(t)'$ after re-normalization.

$$X(t)' = \frac{X(t)}{\sum_{k=0}^{\infty}(X(t))} \tag{18}$$

2) SUM: One of the naive way of implementing SUM function follows closely to Eqn. 1. While, instead of continues distribution, discrete distributions are used for accelerating the calculation process. To find the distribution after the summation operation, the delay time stamp after sum are first calculated. Then, every time stamp are stored into a list. After that, whenever a new time stamp is calculated, the list will be go through to find if there is any match. If so, the product of probabilities of this pair will be added to the accumulated sum for this time stamp. Or if not, this time stamp will be added to the list and probability will be stored if future time stamp match. Although this method yields correct result, it takes too much time for computation. To shorten the time, an improved method has be implemented.

Compare with the naive way of implementation, the input distributions for the improved method are pre-sampled discrete distributions with constant resolution for both inputs. Those input distributions are then stores in numpy arrays for further facilitate the computation. After, that, since now the resolution are fixed for both input distributions, instead of adding them up one by one to find the time stamp after summation, the time stamp after summation can be easily found by adding the smallest time stamp from one distribution to every other time stamp from the other distribution. By doing a sliding window liked operation using pointers, all the combinations that sum up to the resulting time stamp for sum will be included in the overlapping section of the two arrays. Therefore, saves the time for searching through the list each every time in previous

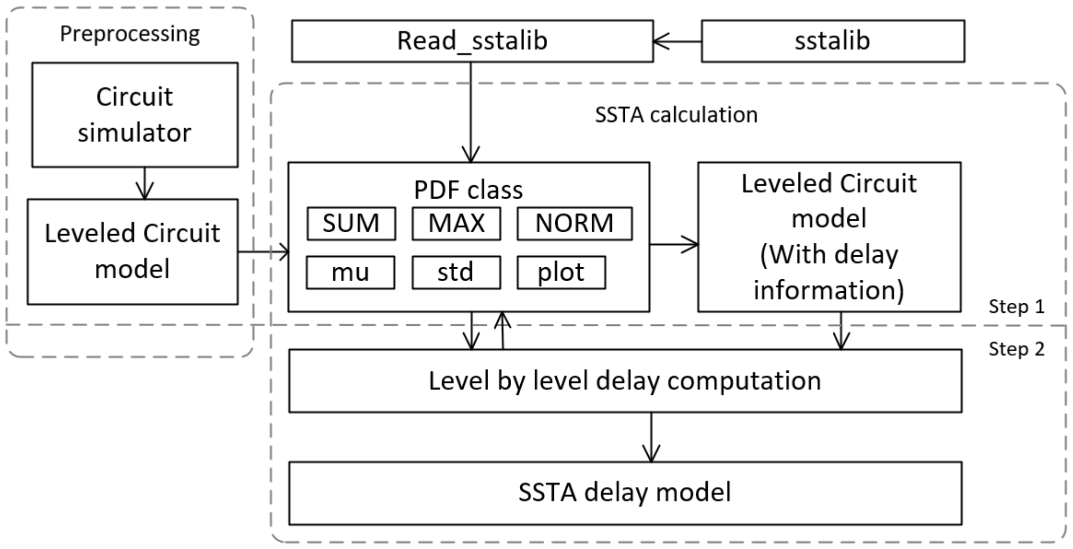


Fig. 8. Overview of Software implementation in block diagram

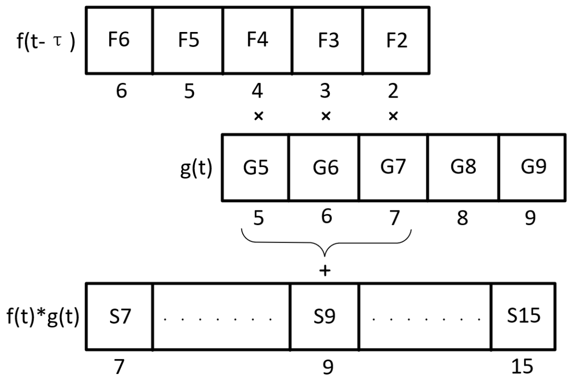


Fig. 9. Illustration of improved SUM

naive way of implementation. Fig. 9, shows a graph illustration for this improved SUM operation, where the constant resolution is 1 for both inputs. Here, the blocks are representing the arrays storing the PDF information denoted with Fi, Gj and Sk, where i,j,k are corresponding values for the time stamp. The value under the blocks are representing time stamp, which also are the delay time. From Fig. 9, one can see, to find the probability for delay time of 9 for the sum distribution, one need to add all the product of probabilities of possible combinations of delay time from f and g that add up to 9, which in this case is exactly the overlapping between this two arrays.

3) MAX: The MAX function takes in two PDF objects. Then, it initializes two numpy arrays, 'max_delay' and 'max_pdf', which stores the delay time stamp and corresponding probability for the max results accordingly. For the max_delay array, to save computational effort, it follows the principle that the minimum value in 'max_delay' array equals to the maximum value between the minimum delay values of the two inputs. While the maximum value in 'max_delay' array equals to the maximum value between the maximum delay values of the two inputs. After getting the range of the 'max_delay' array, the elements in it are then determined by utilizing the constant sampling distance set in the sampling part. With knowing all the delay time stamp for the max result, the max PDF is calculated following the concept stated in Eqn. 10

4) Data_shrink: The 'data_shrink' function is a helper function to accelerate the computation process. It compares the probability value of each PDF object and if the probability is less than a threshold, then its value will be removed from the array. By doing this, it reduces the amount of computations.

5) Dependency List (DL) algorithm: The DL algorithm is implemented to solve the convergence issue in simulation. To do that, a top function called 'reconvergent_top' has been implemented which calls the 'gen_dep_list', 'dep_list' and 'depMax' function. Here, the implementation of 'gen_dep_list" and 'depMax' function basically follows the idea presented in Fig. 6 and 7. Where 'dep_list' is a helper function which find all the nodes for an input of 'i' node depends on using Depth First Search.

IV. Circuit simulation

In this part, both the results of unit test and circuit test will be shown.

A. Unit test

For the unit test, a Python file called 'pdf_exp.py' has been created. This file will take in argument of gate type from command line and will use delay information of input from the 'sstalib'. Then it will generate the delay distribution at the

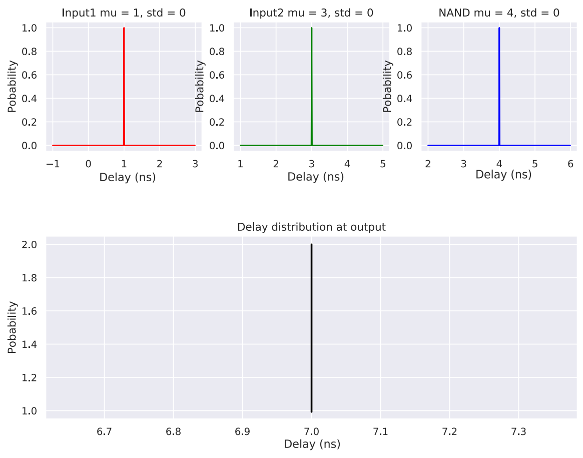


Fig. 10. Unit test with all inputs and gate delay have 0 STD

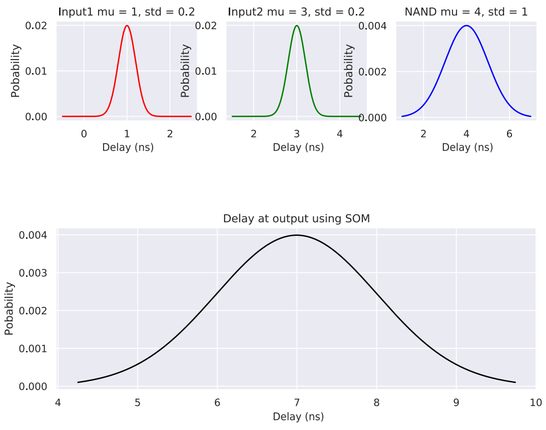


Fig. 11. Unit test with all inputs and gate delay have non 0 STD

output of the assigned gate. Since SUM and MAX function have both been proved in previous section, this test will focus on testing the functionality of SUM of MAX operation, which will be repeatedly used in finding the delay distribution for the proceeding circuit node.

In this test, an NAND gate with two inputs has been selected. The first test will serve as a baseline, where both the inputs to the NAND gate and the NAND gate self delay are set to 0 standard deviation, which makes the SSTA computation the same as STA computation. The result for this test in shown in Fig .10. As now all delays have 0 STD, the maximum between input1 and input2 , which is input2, should be add to NAND delay. Therefore the resulting delay should have a mean value of 7, while the graph shows exactly the same result as expected.

For the second test, the two input delay and the gate delay have been assigned with 0.2, 0.5 and 1 STD value accordingly, while their mean still maintain the same as previous test. The results is shown in Fig. 11

B. Circuit test

For the circuit test, all circuit models are selected from the ISCAS'85 benchmark files from [14]. Here, circuit C17, C432 and C6288 are tested. Where C17 is the circuit shown in Fig. 3. C432 is 27-channel interrupt controller and C6288 is 16x16 Multiplier. Corresponding specifications of these circuit are listed in Table. I, where the number of inverter is not counted in total number of gates. The function used for finding the delay of next circuit node is SUM of MAX, since MAX function saves a bit computation time than SUM function.

For providing a benchmark to test the functionality of the platform, all the delay information has been assigned with zero standard deviation for circuit C17 as if the STA is applied. While all the mean value of each delay set the same as listed in Table. II, the expected delay at each node of C17 with zero STD is shown in Fig. 12. For comparison, Fig. 13 shows the SSTA simulation result for C17 with all delay information the same as Table. II. Since, all given STD are relatively small, the expected result should have a mean value close to the value of STA simulation. As shown in Fig. 13, the mean value is close enough to 20. Thus, prove the functionality of this SSTA simulator.

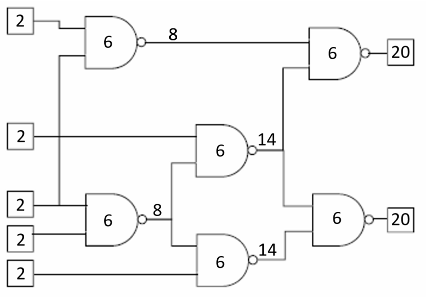


Fig. 12. C17 expected delay for each node if STA applied

For getting a better insight on how complexity of the circuit affect the speed of the platform, a benchmark test with following settings shown in Table. II, is carried out on all three circuits, while assuming all inputs are independent. The time taken for each circuit has been recorded.

According to the test results, C17 takes 1.258s, C432 takes

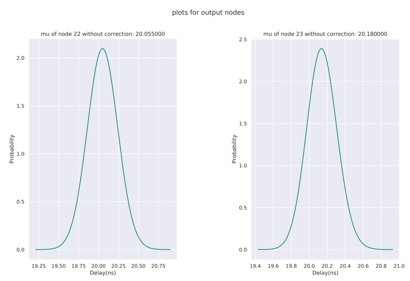


Fig. 13. C17 output results
With Table.II as inputs, assuming all inputs are independent

Table I : Specifications for circuit under test

Circuit	# Inputs	# Outputs	# Inverter	# Gates	#NAND	# AND	# NOR	# XOR
C17	5	2	0	6	6	0	0	0
C432	36	7	40	120	119	4	19	18
C6288	32	32	32	2384	32	256	2128	0

Table II: Circuit test specification I

	Inputs	Inverter	NAND	AND	NOR	XOR
Mean	2	4	6	6.5	7	12
Standard deivation	0.1	0.1	0.1	0.15	0.1	0.2

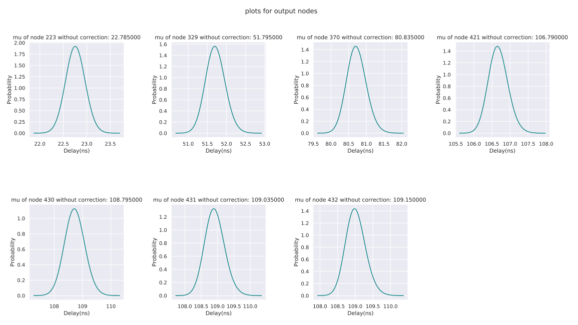


Fig. 14. C432 output results
With Table.II as inputs, assuming all inputs are independent

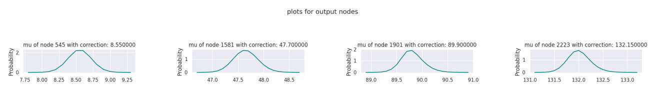


Fig. 15. C6288 output results (Selected)
With Table.II as inputs, assuming all inputs are independent

73.494s, and C6288 takes 148.130s.

For referencing, the results at output of C17, C432 and C6288 are shown in Fig. 13-15. For C6288, only four selected nodes are shown.

After that, the topological correlation is taken into consideration through the dependency list algorithm, same tests are run on C17 and C432 circuits, with input specifications shown in Table. II. The output waveform we get are shown in Fig. 16 and 17. By looking at these two waveform, one can see though there is no significant change in the mean value of the delay distribution, the standard deviation of the output delay has increased after taken the topological correlation into consideration. For a more precised view on the comparison, Table. IV shows the mean and standard deviation of all C432's outputs both before and after the correction, with Table. II as inputs specification. The result shown in table match with the conclusion draw by [4].

To show our results with more inputs variation, tests are then run on input specifications given by Table. III, where we increased both the differences in mean and in standard deviation of the input delay distribution. For this input setting, test are done on C17 and C432 with correlated inputs. Fig. 18 and 19 show the result.

Finally, to verify the correctness of the results, a Monte Carlo (MC) method has been implemented for calculating the delay distribution at the output. Here the Monte Carlo method

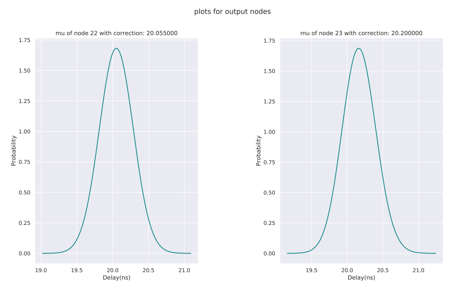


Fig. 16. C17 output results
With Table.II as inputs, considering topological correlation

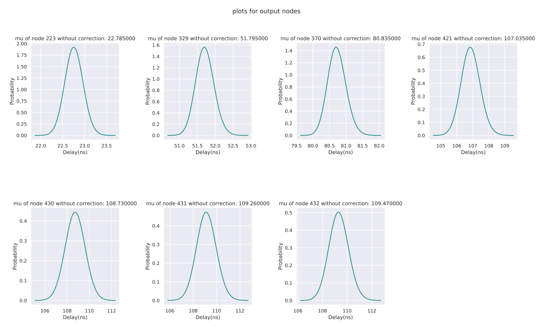


Fig. 17. C432 output results
With Table.II as inputs, considering topological correlation

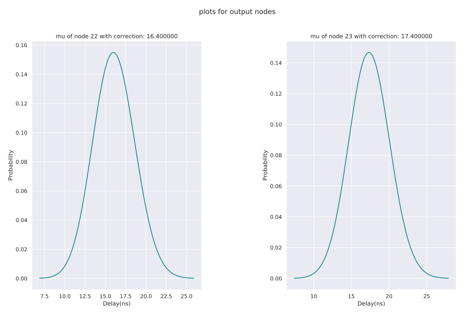


Fig. 18. C17 output results
With Table.III as inputs, considering topological correlation

Table III: Circuit test specification II

	Inputs	Inverter	NAND	AND	NOR	XOR
Mean	2	1	4.5	6	7	12
Standard deivation	1	0.8	1.2	1.5	1.2	2

Table. IV C432 outputs with/without correlation correction, using Table II as inputs

Output Node No.	Without correlation correction		With correlatin correction		Difference in mean (ns)	Difference in STD (ns)
	Mean (ns)	STD (ns)	Mean (ns)	STD (ns)		
223	22.785	0.531	22.785	0.531	0	0
329	51.795	0.652	51.795	0.652	0	0
370	80.835	0.704	80.835	0.704	0	0
421	106.790	0.689	107.035	1.454	0.245	0.765
430	108.795	0.877	108.730	2.092	-0.065	1.215
431	109.035	0.762	109.260	1.988	0.225	1.226
432	109.150	0.707	109.470	1.887	0.320	1.180

Table. V C432 outputs without correlation correction & Monte Carlo method, using Table II as inputs

Output Node No.	Without correlatin correction		Monte Carlo		Difference in mean (ns)	Difference in STD (ns)
	Mean (ns)	STD (ns)	Mean (ns)	STD (ns)		
223	22.785	0.531	22.757	0.416	-0.028	-0.115
329	51.795	0.652	51.584	0.611	-0.211	-0.041
370	80.835	0.704	80.416	0.734	-0.419	0.030
421	106.790	0.689	106.112	0.823	-0.678	0.134
430	108.795	0.877	108.415	0.859	-0.380	-0.018
431	109.035	0.762	108.525	0.830	-0.510	0.068
432	109.150	0.707	108.581	0.812	-0.569	0.105

Table. VI C432 outputs with/without correlation correction, using Table III as inputs

Output Node No.	Without correlatin correction		With correlation correction		Difference in mean (ns)	Difference in STD (ns)
	Mean (ns)	STD (ns)	Mean (ns)	STD (ns)		
223	17.600	5.629	17.300	4.359	-0.300	-1.270
329	46.350	7.159	45.650	5.427	-0.700	-1.732
370	75.350	7.794	74.150	5.716	-1.200	-2.078
421	102.050	8.141	103.250	11.431	1.200	3.290
430	96.700	8.458	95.700	16.021	-1.000	7.563
431	98.850	7.679	99.550	15.945	0.700	8.266
432	99.500	7.419	101.300	15.271	1.800	7.852

Table. VII C432 outputs without correlation correction & Monte Carlo method, using Table III as inputs

Output Node No.	Without correlatin correction		Monte Carlo		Difference in mean (ns)	Difference in STD (ns)
	Mean (ns)	STD (ns)	Mean (ns)	STD (ns)		
223	17.600	5.629	17.398	3.952	-0.202	-1.677
329	46.350	7.159	44.301	5.554	-2.049	-1.605
370	75.350	7.794	71.506	6.852	-3.844	-0.942
421	102.050	8.141	95.747	8.071	-6.303	-0.070
430	96.700	8.458	91.011	8.119	-5.689	-0.339
431	98.850	7.679	92.885	7.818	-5.965	0.139
432	99.500	7.419	102.418	7.770	2.918	0.351

Table. VIII Difference at 3 STD between result before correction and Monte Carlo result with input specification I and II

Output Node No.	Difference at 3 STD (%)	
	Specification I	Specification II
223	-1.55	-17.89
329	-0.63	-11.26
370	-0.40	-7.25
421	-0.25	-5.43
430	-0.39	-5.81
431	-0.28	-4.77
432	-0.23	-3.16

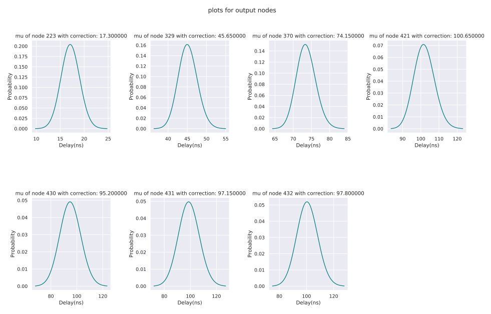


Fig. 19. C432 output results
With Table.III as inputs, considering topological correlation

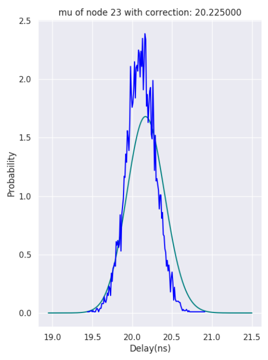


Fig. 20. C17 MC & corrected result for node 23
Blue line is the MC result

implemented does not consider the topological correlation, therefore, only the comparison between Monte Carlo and the result before topological correlation is done. Tests has been done on both C17 and C432 circuit, where Fig. 20, shows an example with the plot of MC result and the result after topological correction for node 23 of C17 circuit. Table. V shows the difference in mean and standard deviation between the result before correction and the MC result using Table II as inputs specification. Similar tests are then done with inputs using Table. III specifications. Corresponding results are listed in Table. VI and VII.

For given a better insight for the result, another comparison has been done on the mean + 3 times standard deviation point between the result before correction and the Monte Carlo result with both input specifications. The difference in percentage has been shown in Table. VIII.

V. Conclusion

In this report, the topic about SSTA has been discussed and a baseline platform has been proposed. Further work could be done on improving the speed of computation and improving the accuracy of the result.

References

[1] J.-H. Liu, A.-S. Hong, L. Chen, and C. Chen, "Process-variation statistical modeling for vlsi timing analysis," 04 2008, pp. 730 – 733.

[2] M. LLC. (1999) MS Windows NT kernel description. [Online]. Available: http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm

[3] "Grinstead and Snell's Introduction to Probability," Tech. Rep., 2006.

[4] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: From basic principles to state of the art," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, 4 2008, pp. 589–607.

[5] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical Timing Analysis for Intra-Die Process Variations with Spatial Correlations," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers*, 2003, pp. 900–907.

[6] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers*, 2003, pp. 621–625.

[7] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical Timing Analysis for Intra-Die Process Variations with Spatial Correlations," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers*, 2003, pp. 900–907.

[8] V. Khandelwal and A. Srivastava, "A general framework for accurate statistical timing analysis considering correlations," in *Proceedings - Design Automation Conference*. New York, New York, USA: ACM Press, 2005, pp. 89–94. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1065579.1065607

[9] A. Devgan and C. Kashyap, "Block-based Static Timing Analysis with Uncertainty," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers*, 2003, pp. 607–614.

[10] S. Tsukiyama, M. Tanaka, and M. Fukui, "A statistical static timing analysis considering correlations between delays," in *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, vol. 2001-January. Institute of Electrical and Electronics Engineers Inc., 2001, pp. 353–358.

[11] C. E. Clark, "The Greatest of a Finite Set of Random Variables," *Operations Research*, vol. 9, no. 2, pp. 145–162, 4 1961.

[12] K. Kang, B. C. Paul, and K. Roy, "Statistical timing analysis using levelized covariance propagation," in *Proceedings -Design, Automation and Test in Europe, DATE '05*, vol. II, 2005, pp. 764–769.

[13] J. Le, X. Li, and L. T. Pileggi, "STAC," in *Proceedings of the 41st annual conference on Design automation - DAC '04*. New York, New York, USA: Association for Computing Machinery (ACM), 2004, p. 343. [Online]. Available: http://portal.acm.org/citation.cfm?doid=996566.996665

[14] "(No Title)." [Online]. Available: http://sportlab.usc.edu/~mitsushij/benchmarks.html