Taylor & Francis
Taylor & Francis Group

Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: 0005-1144 (Print) 1848-3380 (Online) Journal homepage: https://www.tandfonline.com/loi/taut20

Artificial neural network model for arrival time computation in gate level circuits

S. R. Ramesh & R. Jayaparvathy

To cite this article: S. R. Ramesh & R. Jayaparvathy (2019) Artificial neural network model for arrival time computation in gate level circuits, Automatika, 60:3, 360-367, DOI: 10.1080/00051144.2019.1631568

To link to this article: https://doi.org/10.1080/00051144.2019.1631568

Published online: 28 Jul 2019.

AUTOMATIKA
2019, VOL. 60, NO. 3, 360–367
https://doi.org/10.1080/00051144.2019.1631568

REGULAR PAPER

OPEN ACCESS Check for updates

Artificial neural network model for arrival time computation in gate level circuits

S. R. Ramesh [a] and R. Jayaparvathy[b]

[a]Department of Electronics and Communication Engineering, Amrita School of Engineering, Coimbatore, India; [b]Department of Electronics and Communication Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai, India

ABSTRACT

Advances in the VLSI process technology lead to variations in the process parameters. These process variations severely affect the delay computation of a digital circuit. Under such variations, the various delays, i.e. net delay, gate delay, etc., are no longer deterministic. They are random in nature and are assumed to be probabilistic. They keep changing, based on factors such as process, voltage, temperature, and a few others. This calls for efficient tools to perform timing checks on a design. This work presents a technique to compute the arrival time of a digital circuit. The arrival time (AT) is computed using two different timing engines, namely, static timing analysis (STA) and statistical static timing analysis (SSTA). This work also aims to eliminate number of false paths. It uses a fast and efficient filtering method by utilizing ATPG stuck-at faults and path delay faults. ISCAS-89 benchmark circuits are used for implementation. The results obtained using the probabilistic approach are more accurate than the conventional STA. It has been verified with an Artificial Neural Network (ANN) model. The arrival time calculated using SSTA shows 7% improvement over that of STA. The absolute error is reduced twofold in the case of the ANN model for SSTA.

ARTICLE HISTORY
Received 7 January 2019
Accepted 31 May 2019

KEYWORDS
Static timing analysis; statistical static timing analysis; arrival time; false paths; artificial neural network

1. Introduction

Semiconductor devices play a very important role in today's world. Without them, advancements in digital integrated circuits would be impossible. Following technological advancements, it has been possible to mount tens of thousands of semiconductor devices on a small platform. At the same time, the semiconductor process technology shrinks in size. The interconnects that connect various portions of the circuitry have become a reason for reduction of circuit performance in these devices. This is due to the increasing dependency of the devices on the environmental conditions. A few factors may be identified by the designers at the time of design, whereas the prediction of others is very difficult if the design is not complete. Some of them vary during the operation and others are not known for years and are out of the designer's control. The challenge is to manufacture ICs that also function without much trouble in extreme cases. For any digital circuit, knowledge of the delay of the circuit and other timing constraints such as the setup time, hold time, etc., are very important. Hence timing analysis plays a crucial role in VLSI chip design. This process helps to analyse whether a chip design meets the timing constraints or not. Timing analysis mainly focuses on speed and accuracy.

Clock and sequential components are the basis for any timing analysis [1]. For an integrated circuit to function properly, it must meet the time constraints set by the user. Constraints such as setup time and hold time must be verified for every flip-flop in the design. Checking the constraints of the circuit ensures that data which are launched from one flip-flop is captured by another flip-flop without any delays or errors. For digital circuits to be functionally correct, they must run at a certain predefined rate, sequence and specification. Violation of these criteria will affect the system timing and result in errors. Thus, timing analysis is the first operation performed to debug a system and identify the cause of errors.

There are two engines for timing analysis, namely, the static timing analysis (STA) engine and the statistical static timing analysis (SSTA) engine. STA is a method that validates the timing of a circuit without using an input test vector, and the whole circuit need not be simulated. It is implemented such that it is input-independent and finds the extreme or worst-case delay of the circuit by considering all possible input combinations. The delay computation and timing metrics can be obtained by circuit simulation. It is a very time-consuming process and the results are

CONTACT S. R. Ramesh sr_ramesh@cb.amrita.edu Department of Electronics and Communication Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India

based on theoretical assumptions. There is always a trade-off between speed and accuracy in STA. Technological advances in the nanometre regime lead to many parametric variations. Control of the process parameters is very important and increasingly difficult. Process parameters are random in nature. To account for these process parameters, SSTA is preferred. The objective of SSTA is to improve the accuracy without much trade-offs in speed, by considering most of the process variables.

The rest of the paper is organized as follows: Section 2 deals with the background and Section 3 highlights the proposed method. Section 4 depicts the implementation and Section 5 presents the results. Section 6 draws a conclusion.

2. Background and terminologies

The basic terms used in timing analysis are defined as follows:

- A *timing graph* is a directed acyclic graph (DAG) representing a circuit. A circuit is converted into a graph with vertices(nodes) and edges. In timing graphs, the gates of the circuit are represented by vertices, and the interconnects or wires between the gates are represented by edges.

A timing graph G is a graph with a set of vertices V and set of edges E. It is represented as $G = \{V, E\}$ where $V = \{v1,v2,v3, \ldots ,vk\}$and $E = \{e1,e2,e3, \ldots ,en\}$.

Here, each edge e which belongs to E is simply an ordered pair of vertices $e = (vi,vj)$.

- *Arrival time(AT)* at an edge/node in a circuit is the maximum delay for any primary input to the edge/node. It can be also defined as the time that is required for information to travel through the data path.
- *Required time (RT)* at an edge/node in a circuit is the maximum possible delay for any primary output to the edge/node. It can also be stated as the time taken by the clock signal to traverse through the clock path.

STA is deterministic, as it considers all the variables such as process, temperature, voltage, PVT,etc., to be fixed. Hence, the results are more pessimistic. To introduce variations into timing analysis, the gate delays and interconnect delays are assumed to be probabilistic in nature. This improves the timing behaviour of circuits [2]. SSTA has two major approaches, namely, path-based SSTA and block-based SSTA. In the path-based timing approach, the gate delays and the wire delays on the specified paths are added together. Identification of the required path is very important in this technique. There is a high probability that the paths which are relevant in timing analysis may not be selected as the path

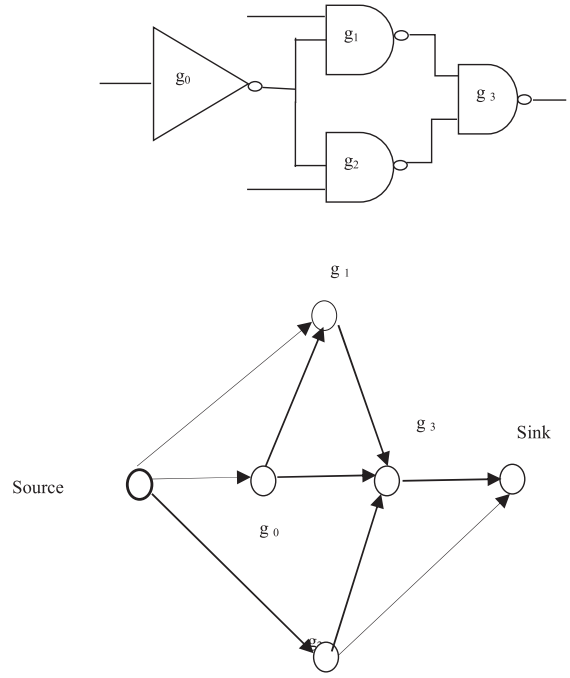

Figure 1. Circuit and Timing graph.

of interest. A circuit and its timing graph are shown in Figure 1. It has one inverter and three 2-input NAND gates. A timing graph is denoted as $G = (V,E)$ where V denotes the vertex and E denotes the edge. The gates are represented as g_0, g_1, g_2 and g_3. Each gate denotes a vertex in the timing graph. Similarly, the edges in the graph are the wires in the circuit. In addition, there are source and sink nodes. All the primary inputs to the graph are connected to the source node and the primary outputs to the sink node. Graph traversal is performed from the source node to the sink node via all possible paths in the graph.

Path selection plays a vital role in path-based SSTA. Here, the timing is performed for selected critical paths only. As the selection of a subset of paths is time-consuming, for large circuits the computational complexity increases with the circuit size. Therefore, path-based SSTA does not give satisfactory results.

To overcome the possible errors due to the path-based approach, a block-based approach has been proposed. By traversing forward and backward in a circuit, the block-based SSTA creates the arrival time and the required time for each of the nodes in the circuit graph. In comparison with path-based SSTA, there is completeness and the path selection is avoided. The main area of concern for block-based SSTA is when statistical maximum operation correlations are to be considered. In block-based SSTA, progressive computation takes place. The timing analysis is performed in the forward or backward direction, block by block for each gate of the timing graph, without considering the path history. Hence, the computational complexity in case of the block-based SSTA changes linearly with an increase in circuit size.

If the probabilistic distributions of the gate and the wire delays are known, the statistical timing behaviour of the target circuits can be improved [2]. False paths represent one of the challenges in SSTA. The issues due to false paths are handled using the path-based approach [3]. In [4], a statistical approach is proposed in which the gate and wire delays are not worst-case examples in linear time. Instead, they are modelled using stochastic values. In [5], statistical delay modelling with gate sizing is proposed. In [6], node delay dependency and its timing analysis are discussed. A probabilistic retiming method is proposed in [7] wherein the circuits are treated as timing graphs and each vertex denotes a combinational element of the circuit with probabilistic characteristics. In [8], the gate delays and arrival time are modelled as discrete random variables. A block-based timing with false path analysis is implemented in [9]. The authors in [10] propose an exact solution that can compute the lower and upper bounds of the delay. A path-based statistical timing approach is proposed in [11]. Here, every path is modelled by the sum of individual gate delays in the path and each gate delay is modelled as a random variable. Then, a linear statistical model for the gate delay is created by taking into account small deviations on either side of the nominal value, and the modelled gate delays are then added together to obtain the path delay. The gate delay and the arrival times are modelled as random variables in [12], where a block-based approach is mentioned in which the gate delays are modelled as PDFs and the arrival times are modelled as cumulative distribution functions (CDFs). Thus, this is efficient and simple for both the SUM and MAX operations.

Developing an appropriate statistical model for the gate delays is the key task in SSTA. To compute the arrival time, the SUM and MAX operators must be used. The delay PDFs are assumed to be probabilistic with normal distributions. When they are directly added(SUM), an error is introduced. Therefore, instead of the statistical SUM, a statistical convolution integral is used. Consider gate delays denoted by delay PDFs f_1 and f_2 that follow normal distributions. Assume m_1 and m_2 denote the mean values and s_1 and s_2 are the sigma values of the delay PDFs. For a cascaded gate connection, the output delay PDF is given by the convolution integral of f_1 and f_2. This also has a normal distribution for the output, with $\mu = m_1 + m_2$ as the mean and $\sigma = \sqrt{({s_1}^2 + {s_2}^2)}$. The MAX operation in statistical static timing analysis is used to compute the delay at the output of a multiple-input gate. The maximum delay of all the primary outputs is obtained and is considered as the required arrival time in SSTA.

The method in [13] provides slack at each node and hence addresses the severity of the timing problem. Several ways to increase the circuit performance and the usage of various delay models are considered in [14]. Various statistical static timing methods and their degrees of focus on accuracy and speed are reported in [15]. A path can also be termed false if the transitions at the input cannot be propagated to the output along the path [16]. STA-reported false and true critical path delay values may or may not be true [17]. Up to 30% of paths in the circuit may be reported as false paths [18]. Therefore, it is predicted that static timing analysis-reported critical paths may be false. This method also neglects simultaneous input transitions. The main reason is that static timing analysis is built on a single input transition (SIT) hypothesis [19].

A. Artificial Neural Network

The most inherent computational model of the human brain is ANN. Its application varies over a wide range from prediction of stock market to pattern recognition. It can use multilayer perceptron and number of hidden layers to solve problems of any complexity. ANNs exhibit parallel architecture [20]. Hence they can be implemented in hardware and software. The size of neural network and its weight adaptation scheme are the major bottlenecks to be decided before implementation.

B. False paths

In the actual functionality of a design, certain paths may not be real. Those paths can be turned off while performing STA, by setting them as false paths. These paths are neglected while STA is carried out. False paths can exist from one clock domain to another, from the clock pin of a sequential element to the input of another element, through a pin of a cell or multiple cells and also via other combinations. When such a path is associated with a pin/port, all paths that go through that particular pin/port are ignored while performing timing checks. If false paths are not identified, a large design space is consumed. Therefore, the focus on real paths is misleading, to a large extent. The analysis time also increases with a large increase in false paths in circuits. Constraints are fed to the tool to set the false paths. A simple false path is shown in Figure 2. The path starting from register B and ending at register C can never be sensitized. This happens when MUX1 selects the output of register B and the combinational logic output is deselected at MUX2.

Path criticality: a critical path can also be regarded as a path which violates the timing constraints of the design. The criticality of a path is the probability of manufacturing a chip in which the path of interest is critical [15,21]. The process of augmenting an existing combinational tool to identify the false paths in sequential circuits is dealt with in [22].

Critical path: the critical path is the maximum path in the circuit which violates the timing constraints.

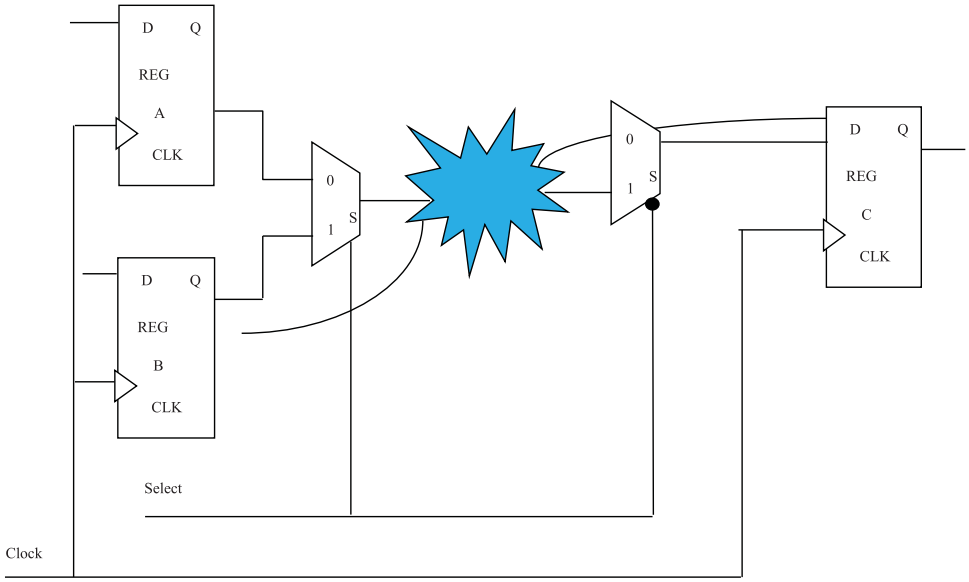

Figure 2. False path in circuit.

Efficient techniques were reported in [23] to identify more false paths using ATPG. STA considers the delay of each gate and logic cell in the propagating path and does not consider the behaviour or functionality of the design [23]. Not all paths will get a chance to be activated in the real circuit. These inactivated paths are referred to as false paths. If any false path is considered as a working critical path, the results are conservative, and the performance of the circuit is degraded. Various techniques associated with slack, false paths and timing issues and constraints are detailed in [24]. ANN based approach is dealt in [25].

3. Proposed method

The primary intention of this work is twofold, focused on refinement in timing analysis and elimination of false paths[26]. The pessimistic nature of STA is explored, and the causes are highlighted. The study first aims to calculate the arrival time of circuits using STA. In addition to this, an SSTA engine is also developed for arrival-time calculation. The computation times for STA and SSTA are also noted. Further elimination of false paths is carried out.

The major contributions of this work are summarized as follows.

(i) Development of STA engine for arrival-time calculation
(ii) Expansion of STA methodology to SSTA formulation for arrival time calculation
(iii) Computation-time calculation in both timing engines
(iv) ANN-based arrival time computation model.
(v) False path elimination in circuits using the ATPG filtering technique.

STA cannot handle process variations, which represents a major drawback. To overcome this, the concept of SSTA is introduced, which represents a major advantage in VLSI. These techniques work on various design corners and the outputs at each stage are modelled with respect to probabilistic distributions. SSTA is not pessimistic in nature, as it takes into account process and temperature variations. The SSTA arrival-time computation follows the procedure shown in Figure 3. Benchmark files of the circuits are read using C code, and the timing graph is generated using the concept of linked lists. The delay library must be extracted from the SDF file and the information is linked to the timing graph. The graph must be levelized with the primary inputs at the first level and the primary outputs at the last level. After levelization, the PDFs of the delay values at each edge in the timing graph are calculated. ANN-based arrival time computation model is utilized to verify the correctness of two-timing engines. Levenberg–Marquardt algorithm is used in this work for the ANN approach.

4. Implementation

The arrival-time computation of the ISCAS-89 benchmark circuits using static timing analysis (STA) and statistical static timing analysis (SSTA) were performed. STA for circuits is described using the flow diagram shown in Figure 3 and the SSTA flow is illustrated in Figure 4. The Verilog file of the circuit must be given to the design compiler. This generates a netlist file for the circuit and a delay file in standard delay format (SDF). The netlist is a description of the connectivity of an electronic circuit. The SDF file contains the delay information for the gates and interconnects.

These files generate the timing report, along with other details like the number of paths, the start point

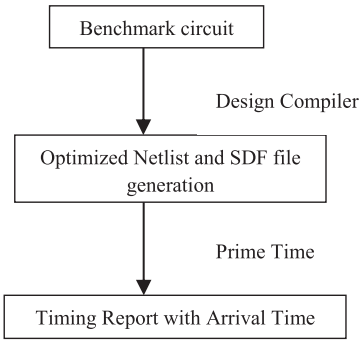

Figure 3. STA flow.

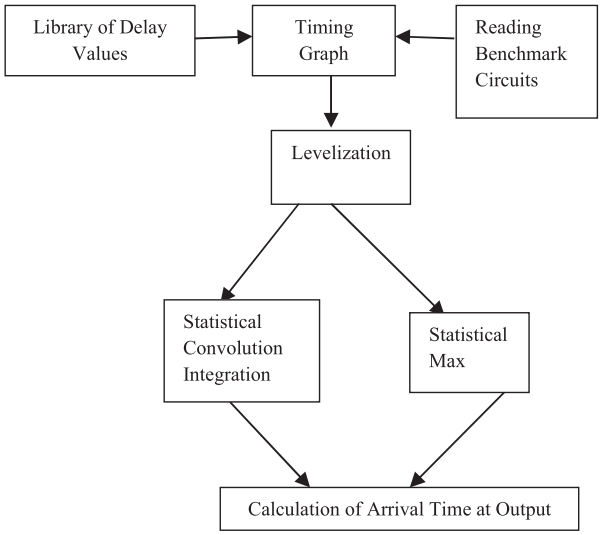

Figure 4. SSTA flow.

and end point of each path and the slack of each path. This is achieved using an STA tool. The tool identifies each path in the circuit and computes the arrival time of every path with the help of fixed information on gate delays. The overall AT is the maximum of the ATs of all path groups.

In the delay library extracted, three delay values, i.e. minimum, typical and maximum values are given. These values are then modelled as normal distributions with a mean and standard deviation at each gate. They are appended to the nodes of the timing graph. The graph is traversed level by level using graph traversal algorithms such as the breadth-first search (BFS) and simultaneously performing statistical operations such as the statistical maximum and convolution integral on each edge and fan-out of the gates of the timing graph. Circuit levelization is generally performed to ensure that the components in the circuit are arranged in an orderly fashion. It is like one component precedes a second component. Hence the evaluation can be carried out in an orderly fashion. In the proposed work levelization is used to perform faster simulations.

Finally, the delay is propagated via graph traversal and the total arrival time is computed at the last level by

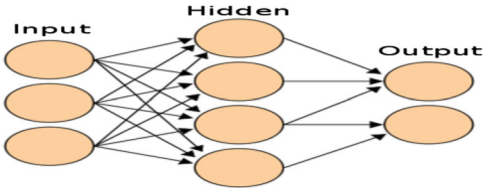

Figure 5. Types of ANN layers.

adding the values from previous levels. Thus, the overall arrival time of the circuit is obtained.

The parameters used to model the ANN are the number of inputs, number of outputs, the gate count, the number of nets, the dynamic power, the circuit levels and the gate count in the critical path. The output of the network is a function of input. It is affected by the weights and transfer function.80% of the samples are used for training and 20% of the samples are used for testing. The various types of layers used in an ANN are depicted in Figure 5. Any neural network will have the following layers: input, hidden and output. The number of hidden layers depends on the type of application. This approach considered one hidden layer and 20 hidden neurons.

The extracted critical paths are converted to TetraMAX [27] readable critical paths. In TetraMAX [27], stuck-at faults are added to all the critical paths to check the functionality of each path. If any path is identified as non-functional, it is detected and set as a false path constraint in PrimeTime [28]. This process of detecting and eliminating false paths is continued until all false paths are eliminated from the circuit. The methodology for false path elimination is illustrated in Figure 6.

Since STA cannot solve the false path problem, a common approach is the use of dynamic timing analysis (DTA). This consumes great deal of time and effort. The optimized gate-level netlist is obtained from the Synopsys Design compiler [29–31]. The netlist thus obtained is imported to the design-for-test (DFT) compiler [32,33]. Scan chain is inserted at this stage and the netlist is extracted for further requirements. ANN and delay based SSTA are reported in [34,35].

5. Results

ANN has the capability of determining the nonlinear relation between the input and the output. Hence it is easier to estimate the output predicted using both timing engines. Table 1 shows the results for levelization of the timing graph. The primary inputs, primary outputs and levels are identified based on the node properties.

Table 2 shows the arrival times for STA and SSTA.

It is evident from Table 2 that the arrival times for SSTA are lower than for STA. To achieve effective functioning of the circuit, the clock may have to be slowed down in many cases.

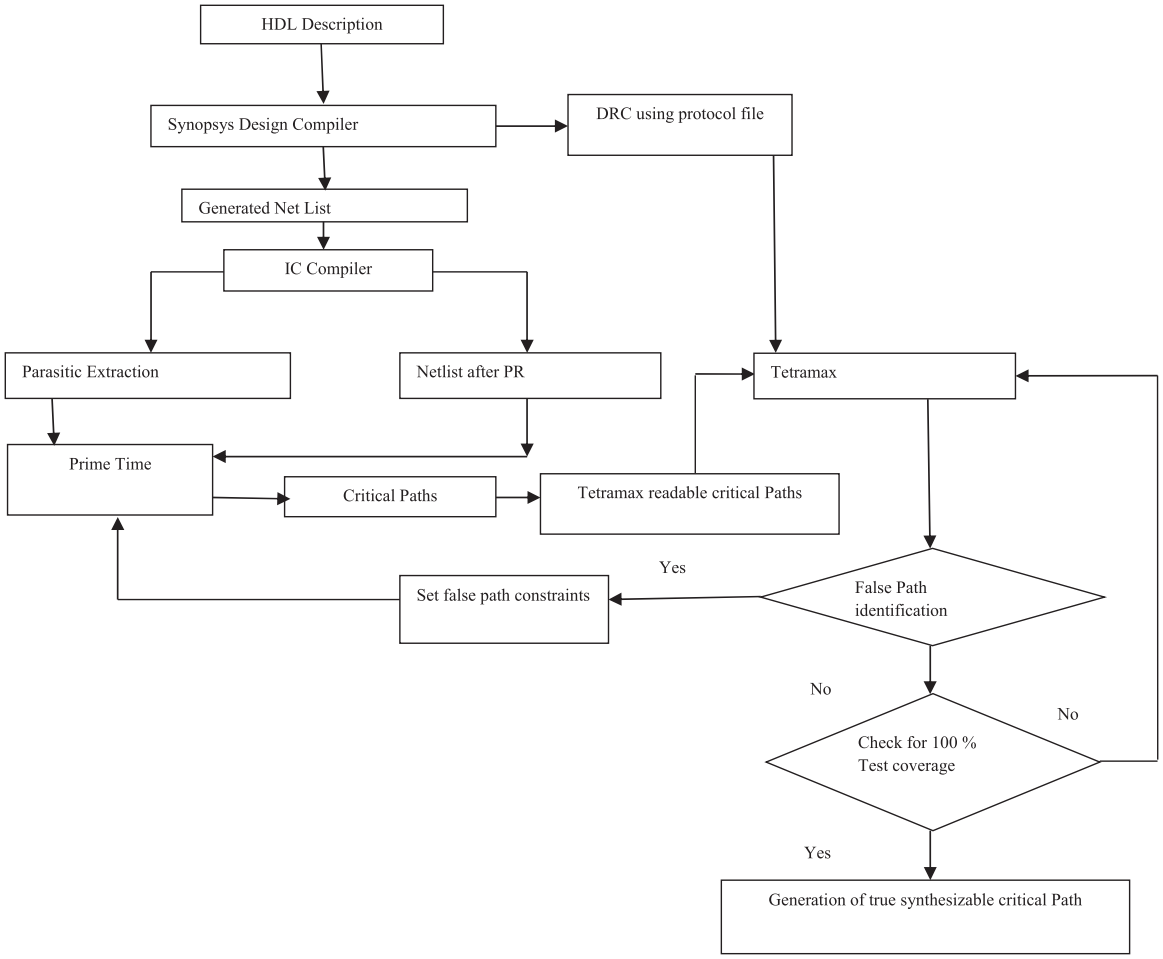

Figure 6. False path elimination method.

Table 1. ISCAS-89 benchmark circuits – primary inputs, primary outputs and levels.

Benchmarks	Primary inputs	Primary outputs	Levels
S400	3	6	7
S820	18	19	9
S832	18	19	9
S1196	14	14	15
S1238	14	14	8
S1423	17	5	9
S1488	8	19	12
S5378	35	49	21
S9234	19	22	29
S13207	31	121	37
S15850	14	87	44
S35932	35	320	15

Table 2. Arrival time for STA and SSTA.

Benchmarks	STA (ps)	SSTA (ps)
S400	198	190.2
S820	322	310.4
S832	376	344.2
S1196	616	579.4
S1238	657	633.1
S1423	702	675.2
S1488	799	772.13
S5378	947	940.78
S9234	1106	1093.87
S13207	1379	1361.05
S15850	1685	1498.24
S35932	501	307.49

Table 3. Computation time for STA and SSTA.

Benchmarks	STA(s)	SSTA(s)
S400	0.000^{+}	0.000^{+}
S820	0.000^{+}	0.000^{+}
S832	0.001	0.001
S1196	0.010	0.020
S1238	0.010	0.020
S1423	0.010	0.010
S1488	0.010	0.020
S5378	0.270	0.280
S9234	0.870	0.940
S13207	1.730	1.850
S15850	2.530	2.680
S35932	8.730	10.02

Consider the circuit S1488. The arrival time using STA was 799 ps and that using SSTA was 772.13ps. On average, the arrival times estimated using SSTA showed a 7% improvement over the STA values. Thus, the pessimistic nature of STA is proved.

Table 3 shows the computation time for STA and SSTA. The computation time for SSTA is greater than for STA in most cases. The experiments were carried out on a 3.30 GHz Intel Core i5-2500 processor.

It is depicted from Figure 7 that the arrival times computed using SSTA engine with ANN provides less absolute error than STA. Hence it is proven that SSTA results are valid and matches closely with the actual model.

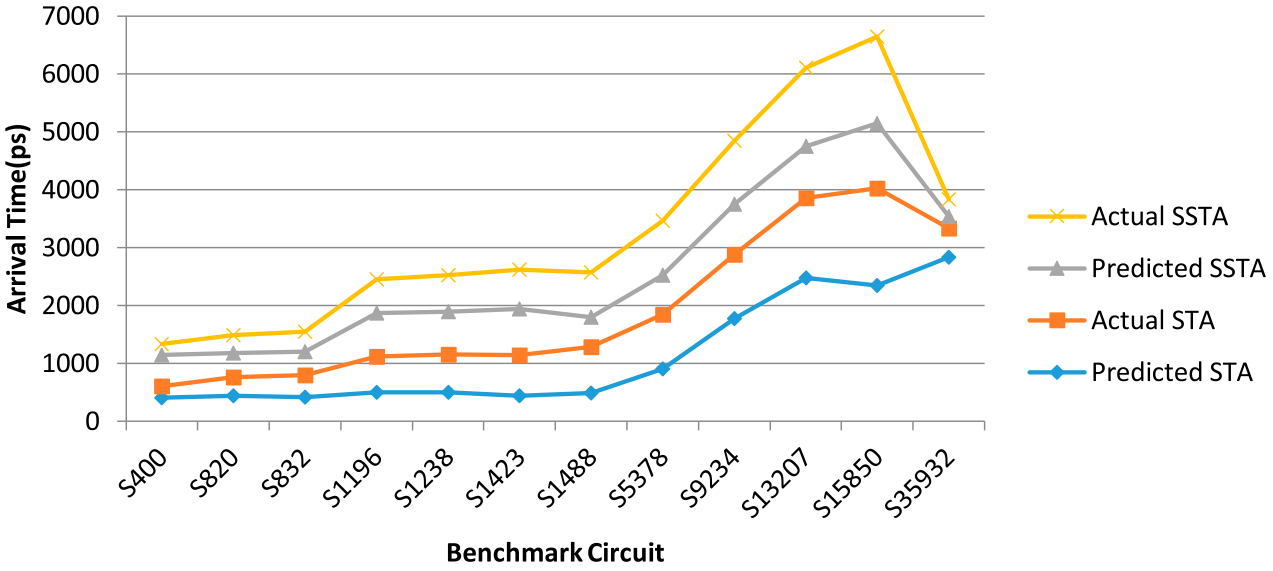

Figure 7. Arrival Time with and without ANN.

Table 4. Number of critical paths before and after filtering technique.

Design	Total critical paths (FFs)	Total false critical paths identified (FFs)	STA critical pathdelay (ns)	True Synthesizable critical path delay (ns)
S400	21	10	1.15	1.12
S820	16	0	2.90	2.90
S832	32	0	1.89	1.89
S1196	18	3	2.03	1.96
S1238	18	4	1.77	1.74
S1423	74	50	5.22	4.87
S1488	6	2	1.66	1.60
S5378	179	13	1.59	1.55
S9234	211	15	2.07	2.03
S13207	638	52	2.85	2.79
S15850	534	105	4.36	4.25
S35932	1,728	28	13.39	13.20

Table 5. STA true critical path delays with STA false critical path delays.

Design	STA true critical path delay (ns)	STA critical path delay (ns)
S400	1.14	1.15
S820	2.90	2.90
S832	1.89	1.89
S1196	2.02	2.03
S1238	1.76	1.77
S1423	5.22	5.24
S1488	1.66	1.66
S5378	2.05	2.07
S9234	2.63	2.68
S13207	2.54	2.85
S15850	4.35	4.36
S35932	13.39	13.39

Table 4 shows the total critical paths and the total false critical paths. STA-reported critical paths with path delays and the true synthesizable critical path with path delay after filtering of false paths are also reported. In S832 and S820, no false path is reported; hence, the STA critical path is the true critical path. Table 5 shows true critical path delays and false critical path delays.

6. Conclusion

In this paper, several ways to perform an efficient timing analysis were explored and the arrival times for ISCAS-89 benchmark circuits were obtained. The STA and SSTA arrival times are calculated and compared. It is seen that the values of SSTA are lower than the corresponding STA values. STA and SSTA arrival times are compared with the ANN model. It is proved that the ANN model outperforms the SSTA computation and produces less absolute error. Static timing analysis reports the critical paths and false paths which are not accurate, and the reported critical paths may or may not be synthesizable. In addition, static timing analysis cannot identify the number of false critical paths in the circuit. Dynamic simulation is the solution for false path identification. The proposed methodology identifies the number of false paths in the circuit through ATPG, and false paths are eliminated through set false-path constraints.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

S. R. Ramesh http://orcid.org/0000-0003-4961-0208

References

[1] Devadas S, Jyu H, Keutzer K, et al. Statistical timing analysis of combinational circuits. Proceedings of IEEE International Conference on Computer Design, October 1992. p. 38–43.

[2] Jyu H, Malik S. Statistical timing optimization of combinational circuit. Proceedings of IEEE International Conference on Computer Design, October 1993. p. 77–80.

[3] Brashear RB, Menezes N, Chanhee O, et al. Predicting circuit performance using circuit level statistical timing analysis. Proceedings of European Design and Test Conference, March 1994. p. 332–337.
[4] Berkelaar M. Statistical delay calculation: a linear time method. Proceedings of International Workshop on Timing Analysis TAU. p. 1585–1588.
[5] Jacobs E, Berkelaar M. Gate sizing using a statistical delay model. Proceedings of European Design and Test Conference, March 2000. p. 283–290.
[6] Lin R, Wu M. A new statistical approach to timing analysis of VLSl circuits. Proceedings of International Conference on VLSl Design, January 1998. p. 507–513.
[7] Tongsima S, Chantrapomchai C, Sha EHM, et al. Optimizing circuits with confidence probability using probabilistic retiming. Proceedings of International Symposium on Circuits and Systems, June 1998. p. 270–273.
[8] Liou JJ, Cheng KT, Kundu S, et al. A fast statistical timing analysis by probabilistic event propagation. Proceedings of ACM Design Automation Conference, June 2001. p. 661–666.
[9] Liou JJ, Krstic A, Wang LC, et al. False path aware statistical timing analysis and efficient path selection for delay testing and timing validation. Proceedings of ACM Design Automation Conference, August 2002. p. 566–569.
[10] Agarwal A, Zolotov V, Blaauw D, et al. Statistical timing analysis using bounds. Proceedings of Design, Automation and Test in Europe, March 2003. p. 348–353.
[11] Orshansky M, Keutzer K. A general probabilistic framework forworst-case timing analysis. Proceedings of ACM Design Automation Conference, June 2002. p. 556–561.
[12] Devgan A, Kashyap C. Block based timing analysis with uncertainty. Proceedings of ACM Design Automation Conference, November 2003. p. 607–614.
[13] Hitchcock RB. Timing verification and the timing analysis program. Proceedings of Design Automation Conference, February 2006. p. 594–603.
[14] Jouppi NP. Timing analysis for nMOS VLSI. Proceedings of Design Automation Conference, February 2006. p. 411–418.
[15] Ramesh SR, Jayaparvathy R. Improved statistical static timing analysis using refactored timing graphs. J CTN. November 2016;13(11):8879–8884.
[16] Jongyoon J, Kim T. Variation-aware false path analysis based on statistical dynamic timing analysis. IEEE Trans Comput Aided Des Integr Circuits Sys. October 2012;31(11):1684–1697.
[17] Wang SJ, Tzeng TH, Li KSM. Fast and accurate statisticalstatic timing analysis. Proceedings of International Symposium on Circuits and Systems, July 2014. p. 2555–2558.
[18] Jun X, Li X. Improve accuracy of delay element by filtering false path for low power desychronized circuits. Proceedings of International Symposium on Circuits and Systems, July 2011. p. 845–848.
[19] Tsai S, Huang CY. A false-path aware formal static timing analyzer considering simultaneous input transitions. Proceedings of Design Automation Conference, August 2009. p. 25–30.
[20] Mohankumar N, Bhuvan B, Nirmala Devi M, et al. A modified genetic algorithm for evolution of neural network in designing an evolutionary neuro-hardware. Proceedings of International conference on genetic and evolutionary methods, July 2008. p. 108–111.
[21] Crouch AL, Potter JC. Invited-A box of dots: using scan-based path delay test for timing verification. Proceedings of Design Automation Conference, August 2016. p. 1–6.
[22] Bell JL, Sakallah K, Whittemore J. False path analysis in sequential circuits. In 8th International Workshop on Power and Timing Modeling, Optimization and Simulation, August 2007. p. 1–10.
[23] Zeng J, Abadir M, Abraham J. False timing path identification using ATPG techniques and delay-based information. Proceedings of Design Automation Conference, June 2002. p. 562–565.
[24] Parnerkar SV. Timing false path identification using ATPG techniques [MS Dissertation]. University of Wisconsin.
[25] Nair BB, Kumar PN, Prasad SR, et al. Forecasting short term stock prices using sentiment analysis and artificial neural networks. J Chem Pharmac Sci. March 2016;9(1):533–536.
[26] Bhasker J, Chadha R. Static timing analysis for nanometer designs: a practical approach. Springer Science Business Media; 2009.
[27] Synopsys Inc. (2014). HDL compiler for verilog user guide version J – 2014.09, September 2014.
[28] Synopsys Inc. (2015). PrimeTime user guide version K –2015.12, December.
[29] Synopsys Inc. (2016). IC compiler implementation user guide version L – 2016.03, March 2016.
[30] Synopsys Inc. (2016). Design compiler user guide version L – 2016.03, March 2016.
[31] Synopsys Inc. (2016). TetraMAX ATPG user guide version L –2016.03-SP1, April 2016.
[32] Synopsys Inc. (2016). DFT compiler, DFTMAX, and DFTMAX ultrauser guide version L –2016.03, March 2016.
[33] Synopsys Inc. (2015). StarRC user guide and command reference version K –2015.12, December 2015.
[34] Das BP, Amrutur B, Jamadagni HS, et al. Voltage and temperature-aware SSTA using neural network delay model. IEEE Trans Semicond Manuf. Nov. 2011;24(4):533–544.
[35] Freeley J, Mishagli D, Brazil T, et al. Statistical simulations of delay propagation in large scale circuits using graph traversal and kernel function decomposition. 2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Prague, 2018. p. 213–216.