Formal Timing Analysis of Gate-Level Digital Circuits using Model Checking

Qurat-ul Ain[a,*], Osman Hasan[a]

[a]School of Electrical Engineering and Computer Science National University of Sciences and Technology (NUST) Islamabad Pakistan

ARTICLE INFO

Keywords:
Formal Verification
Uppaal
Timed Automata
Model Checking
Timing Analysis

ABSTRACT

Due to the continuous reduction in the transistors sizing ruled by the Moore's law, digital devices have become smaller, and more complex resulting in an enormous rise in the delay variations. Therefore, there is a dire need of precise and rigorous timing analysis to overcome anomalies during the timing analysis. Timings of digital circuits can be verified using various simulation or static timing analysis (STA) based tools but they provide estimated results due to their inherent in-exhaustive nature or report timing paths corresponding to non-existent functional paths, respectively. Formal verification provides complete and sound analysis results and has widely been used for the functional verification of digital circuits but its application in the timing analysis domain is somewhat limited. We present a generic framework to perform formal timing analysis of digital circuits with the help of Uppaal model-checker. The given digital circuit along with its timing parameters in the form of state transition diagram are modeled using timed automata in the Uppaal model checker. Timing delays are calculated from corresponding technology parameters, and Quartus Prime Pro is used to obtain the information about the circuits' paths. In order to make the analysis scalable, we also propose a novel path partitioning technique and compare its results with complete path analysis and traditional STA. The formal model is verified with the help of properties to assess the timing characteristics, like time period of a clock, critical path, and propagation delay of the considered circuit. Modeling and verification of ISCAS-85 and ISCAS-89 benchmark circuits is presented for illustration purposes.

1. Introduction

Due to the increased complexity in integrated circuits, modeling and verification of both functional and non-functional characteristics of digital circuits have become very difficult. In timing analysis, our objective is to determine the delay of each component of a circuit based on the underlying physical technology while considering the on-chip physical [5] variations, like gate-oxide thickness, doping concentration, and channel length. These physical variations result in variations in the electrical parameters, such as gate capacitance, resistance and capacitance of wires, and threshold voltage which in turn result in an uncertain delay for digital components. The overall delay of a circuit is calculated by simply accumulating delays of all of its components using several techniques, like static timing analysis [31] or gate level simulations [39]. Static timing analysis (STA) [12] is more scalable, compared to gate-level simulation, and thus a more widely adopted technique since 1990s. The STA is carried out statically, i.e., the analysis results do not depend upon input data values. The purpose of STA is to verify if the given design can operate safely at the given speed without any timing violations. However, complex digital circuits cannot be exhaustively verified by STA due to the large number of associated physical variations and thus the analysis is based on either considering the worst case scenario or a sampled subset of all the possibilities. This kind of a worst case or an incomplete analysis can produce a non-optimal design in terms of timings, which is a very undesirable feature given the safety, performance and security-critical nature of domains in which digital designs are used.

Recently, very powerful and performance-efficient open-source STA tools have been developed, including iTimerC [32], and OpenTimer [28, 29]. An efficient Graph-based timing propagation (GBP) framework is proposed [21, 27] by exploring both structural and pipeline parallelisms in the STA task graph. These tools can analyze circuits with a considerably large number of gates efficiently but these STA based techniques do not consider the functionality of the circuits and the impact of all the possible input combinations on the delays.

Formal verification [24] is known to overcome the limitations of simulation based approaches. It has been broadly advocated for the digital circuits' functional verification [9, 15, 30, 40]. Formal verification mainly involves constructing a formal model and verifying its desired behavior based on its corresponding formal specifications. One of the most frequently used techniques of formal verification is a model checking. It provides automatic verification and can generate a counter example in case if a property is not successfully verified. In model checking, state transition diagram is mainly used for modeling the given system and this state space is explored exhaustively to check if a desired property holds for the given model.

In recent years, formal timing verification of integrated circuits got considerable attention as timing analysis of digital circuits plays a critical role in the overall performance of a device. The timing behavior of combinational circuits has been analyzed using the model-checker Open-Kronos [38]. Small combinational circuits with several gates have been verified using the timed automata (TA) along with some

*Corresponding authors.
Email addresses: qain.msee15seecs@seecs.nust.edu.pk (Q. Ain); osman.hasan@seecs.nust.edu.pk (O. Hasan)

abstraction techniques. Stabilization time of some basic circuits has been analyzed using the timed automata. In [13], a digital circuit is divided into smaller sub-circuits using the reachability graphs, and modeled and verified using the timed automata. A major drawback in this technique is that it does not use the actual delay values for the logic gates and instead it uses constant delays, e.g., the inverter delay is considered to be 0, and thus the technology parameters are fully ignored in this approach. Formal verification of both combinational and sequential circuits is performed in [10]. Circuit behavior is modeled in two different views i.e., functional view, and a timed view. The functional view is associated with a synthesizable hardware description language VHDL, while the timed view is described in terms of propagational delays of circuit's components, which is obtained from SPICE simulations. These two views are modeled in timed automata to verify response time, stability period, and the event delays. Linear constraints are formally derived to analyze the performance of SPSMALL memory in [20]. These constraints are derived using the parametric timed automata [14] and the HYTECH tool [26] to guarantee the correct behavior of a circuit. The input signal propagates to the output port after passing through a certain number of logic components. Each logic component has delays at rising and falling edges. Model based on parametric timed automata has been developed for the verification of SPSMALL memory architecture [20]. In this model, set of linear constraints have been derived to ensure the correct response time of a memory architecture. Similarly, timed circuits are formally verified with the unspecified symbolic delays in [22]. Symbolic delays of each event is defined in an interval $[d_i, D_i]$ where d_i represents the minimum delay value and D_i symbolizes the maximum delay value. This technique used an algorithmic approach for linear constraints in timed transition system (TTS) to verify timed circuits but this approach is limited to verification of circuits with upto a maximum of 20 symbolic delays and therefore is suitable for smaller size circuits only.

Digital circuits have also been formally verified with respect to time for numerous other applications e.g., fault detection in the circuits [42], and detection of Hardware Trojans (HT) [2]. Propagational delays are modeled in an interval $\delta[\tau_{min}, \tau_{max}]$ for verification of synchronous sequential circuits in [42] where $\delta[\tau_{min}]$ represents the minimum delay value and $\delta[\tau_{max}]$ represents the maximum delay value in a considered circuit. Circuits are first modeled as a timed automata (TA), faults are then injected in circuits and verified against certain set of properties. Counter examples obtained after verification play a critical role in fault detection. Similarly, combinational circuits has been formally verified in [2] and this approach has been used for Hardware Trojan detection using the gate-level side channel parameters, like power and delay. Intrusion is injected in a given circuit in terms of logic gates. Generation of counter examples from the model-checker provide the basis of intruded paths. This approach is limited to the verification of combinational circuits only. Furthermore, the paths of combinational circuits are analyzed manually for the verification of propagational delay. Formal verification of integrated circuits (ICs) is performed in [3] for analyzing vulnerabilities in ICs using gate-level side channel parameters. Combinational circuits are modeled with the help of nuXmv model-checker to evaluate the safe range of propagation delay, switching power and leakage power. In case of vulnerabilities, the values of power or the delay exceeds from the given range, and thus the counter example is generated during LTL property verification. This technique is used to verify the combinational circuits only. Moreover, with the increase in the circuit size, intrusion detection time increases exponentially. That is why, this technique is limited to the verification of circuits with a limited number of gates i.e., the maximum circuit that was verified was composed of 1669 gates. A model checking based methodology, based on nuXmv [18], for security vulnerability analysis is presented in [4]. In this approach, safe bounds of power and delay are calculated using Matlab. 3-Sigma process variation is also modeled considering threshold voltage V_{th}, thickness oxide t_{ox}, and effective channel length L. This technique has scalability issues and is applicable to small circuits only i.e., the largest verified circuit had 166 gates, and it performs verification of the combinational circuits only. This approach models gate-level side channel parameters using nuXmv model-checker. In [41] circuit is modeled as a network of stochastic hybrid timed automata using statistical model checking approach. This technique generates fault coverage against the modeled circuits. By carefully analyzing the counter examples, it optimizes the inputs vectors in order to minimize the deviation from the desired timing behavior. This approach modeled both the combinational and sequential circuits but all the timing analysis is based on modeling of delay as a random variable instead of realistic value. Accurate computation of circuit delay using timed automata is presented in [1]. Circuits are modeled as network of timed automata and delay is computed by performing a symbolic traversal of the state space. This technique can only verify small size combinational circuits i.e, upto 16 bit carry skip adder. Accurate and approximate logical multipliers are formally modeled using basic gates in the UPPAAL SMC model-checker and their error rate is analyzed against different inputs in [34]. The approximate multiplier is more efficient in terms of area and performance because it has less number of logic gates than the accurate multiplier and hence it requires less states for its modeling and verification. Similarly, model-checking approach has been used to analyze the security of SoCs from external threats in [25]. The state transition diagrams are modeled using the gate-level netlists and verified against various properties by analyzing possible attacks from Trojans in bus protocol and IPs. Both of these approaches utilize model checking based formal verification of digital circuits by formally modeling them at the gate level but they do not focus on critical path analysis and delay computations of logic circuits, which is the context of the current paper.

In this paper, we overcome the limitations of above mentioned approaches to propose a more accurate formal timing

analysis technique. Digital circuits, comprising of both combinational and sequential parts, are formally verified using the model-checker Uppaal in [7] and delay computation has been achieved using the Elmore delay model [45]. Furthermore, rather than making the use of bi-bounded delays, we proposed to perform the delay computations at each possible input combination of each gate present in the digital design. For example, delays are computed for all the four possible transitions $\Gamma_{Delay} = [d_{00}, d_{01}, d_{10}, d_{11}]$ in case of a two input logic gate where d_{00} symbolizes the delay value when both inputs of a gate are 0. Similarly, delay values against inputs 01, 10, 11 are represented as d_{01}, d_{10}, d_{11}, respectively. Moreover, instead of exploring the timing paths manually within a circuit, we propose to extract the paths of a given circuit automatically using the Quartus Prime Pro software [35]. However, the approach was not scalable as with the increase in circuit size, the respective state space and thus the verification requirements, in terms of time and memory, grew exponentially. In this paper, we introduce path based modeling and verification technique using the model-checker Uppaal to overcome the above-mentioned scalability issues. The largest verified circuit using this approach consists of 2779 gates, which is better than the above-mentioned existing approaches.

The proposed approach introduces some pessimism during circuit modeling and verification and presents a new trade-off between STA and traditional formal timing analysis. This pessimism mainly arises by splitting longer paths in two or more parts and modeling each part as an independent Uppaal template. Inputs of each part are modeled non-deterministically and hence results in over-estimated delay values. Modeling of all the possible input combinations for off-path logic also results in pessimistic delay values. This approach allows us to reduce the verification time and memory utilization significantly at expense of slight pessimism in circuits' delay. However, it is important to note that despite these pessimisms, the obtained delay values are much more realistic than the ones obtained. from pure STA as will be demonstrated in the Case study section of this paper as well. This technique also reduces the state space for digital circuits compared to [7] remarkably. Thus, the proposed approach is able to verify significantly larger number of combinational as well as sequential circuits (approximately 36 times larger compared to our previous work). To assist the process of modeling and verification, a generic framework is provided in which using the information of delays of basic logic elements i.e., NOT, NOR, NAND, and a Flip-Flop, timing behavior of any given digital circuit can be verified such as the critical paths, the clock time period of a circuit, plus setup and hold time constraints of a circuit. Since we are using model checking for timing analysis, so the proposed approach involves rigorous exploration of state space of a considered digital circuit.

The remaining paper is structured as follows: We present some preliminaries, including the foremost foundations about timing analysis, performance parameters and the Uppaal model-checker in Section 2. The detailed explanation of the proposed methodology is presented in Section 3, followed by the verification results of some real-world case studies in Section 4. Finally, Section 5 concludes the paper.

2. Preliminaries

2.1. Timing Analysis

Propagation delay of a digital circuit is the most important timing parameter that represents the time taken by a digital signal to proceed from its input port to the output port. Propagation delay can be calculated using various techniques i.e., using the SPICE simulator, mathematical analysis based on differential equations, and the analysis of rise and fall times. In SPICE [33], the circuit is modeled at the transistor level. Internal specifications of transistors are included in the model using the predictive library parameters. Simulations of transistor level circuits with the help of SPICE is used to calculate the propagation delay of a given circuit. Similarly, the circuit behavior can be modeled as a set of complex differential equations in terms of the resistances and capacitances of the underlying transistors. This mathematical model can then be used for the propagation delay calculation by carefully observing the response of the circuit at the given conditions. The rise and fall times based timing analysis technique is most commonly used for the calculation of propagation delay. In this technique, the propagation delay is typically calculated on the basis of time difference in such a way that the time from middle point of the input to the output signal middle point is measured. This delay is usually considered to be 50% point of the input-output transition.

$$T_{prop} = \frac{T_{HL} + T_{LH}}{2} \tag{1}$$

In the proposed methodology, the particular gate delays are calculated by considering the transition at each and every input combination of a gate with the help of the Elmore delay model [2], which allows us to find the delay by illustrating every individual circuit in the form of an RC (resistance-capacitance) tree and hence provides a better estimate of the delay compared to the above-mentioned traditional approach [19]. We have used Elmore delay because of its comparative simplicity and closely approximating the actual delay values in terms of net widths and lengths [23]. These benefits have led to its wide usability for delay predictions in the placement and routing phases of physical design. If C_i is the accumulated capacitance, and R_{is} is effective resistance present in a path from a source to a leaf node, then the Elmore delay T_e of this path will be,

$$T_e = \sum_i C_i \times R_{is} \tag{2}$$

$$\tau_{delay} = T_e \times ln(2) \tag{3}$$

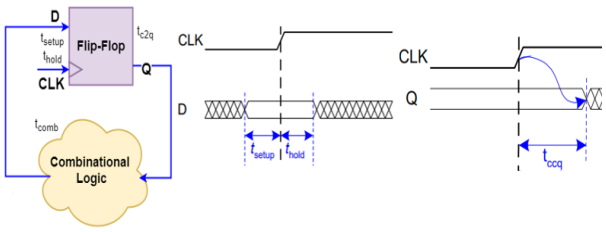


Figure 1: T_{setup}, T_{hold}, and T_{Clk2Q} Delay in a Timing Diagram

It is important to note that although we used the Elmore delay model for delay calculations, the proposed approach is not restricted to only Elmore delay. We can use other delay models with the proposed approach as well if required.

Some other important parameters that determine the timing behavior of circuits include critical path, setup and hold time constraints, clock time period, e.t.c. Critical path is the longest path in a circuit from the input port to the output port. It can be obtained by adding the individual delays of all the logic components present in the longest path of the circuit. Timing analysis in a sequential circuit is more complex because each bit works in synchronization with a clock. The significant timing parameters in a sequential circuit include setup time, hold time and clock to Q delay [44], as shown in Figure 1. The time interval before the clock edge at which the data must be stable is called the setup time T_{setup}. The minimum time interval at which data should be stable after the clock edge is the hold time T_{hold}. Clock to Q delay T_{Clk2Q} is the maximum time from the clock to the output port. Setup time and hold time violations can be avoided by meeting the following conditions where T_{comb} is the total delay of all the combinational elements in a particular path.

$$T_{Clk2Q} + T_{comb} + T_{setup} \leq T_{clkperiod} \tag{4}$$

$$T_{Clk2Q} + T_{comb} \geq T_{hold} \tag{5}$$

2.2. Uppaal Model-Checker

Uppaal [11] is a model-checker, based on the timed automata theory [8], developed for the verification of real-time systems.

A timed automaton (TA) can be defined as a tuple $TA = (S, s_o, \Gamma, \sigma, Y, \beta)$, where:

1. S represents the set of all the locations present in a TA.
2. $s_o \in S$ depicts the initial location of a TA.
3. Γ represents the set of all the clocks declared in TA.
4. σ shows the set of all the actions in a TA.
5. $Y \subseteq \sigma \times B(\Gamma) \times S \times 2^{\Gamma} \times S$ is the set of all the edges present among the various locations.
6. $\beta : S \rightarrow B(\Gamma)$ represents the invariants assignment to the particular locations.

$B(\Gamma)$ shows the set of conjunctions over some simple conditions, e.g., $x \bowtie c$ or $x - y \bowtie c$, where $x, y \in \Gamma, c \in \mathbb{N}$ and $\bowtie \in \{\geq, =, \leq, <, >\}$. A clock valuation is a function $u : \Gamma \rightarrow \mathbb{R}_{\geq 0}$ from the set of clocks to the positive real values. Thus, u satisfies $\beta(s)$ can be written as $u \in \beta(s)$. Timed automaton is basically a finite state automaton having set of transitions and states, enriched with built-in real value clocks, which evolve at a fix rate and can be modeled to reset to their initial value.

A state can be defined as a set of pair (S, α), where α shows the value of variables and clocks exist in that specific state. A state has a discrete transition t, and if the guards associated with the particular transition t are satisfied then the system moves to the next state (S', α'). Guards are primarily the constraints that are present on a transition t, and allow the system to move to the next state. The synchronization channels are used for interconnection between two or more timed automata. The synchronization signal is sent by an automaton in a desired transition t and received by one or more automata.

Model verification using the distinct properties is a critical step in a model-checking. Just like the system model, system properties can be written in a formal language. Uppaal supports a simplified version of TCTL (timed computational tree logic) properties.

The Uppaal model-checker is widely used for modeling and verification of real-time systems, which are modeled as timed automata. The main idea is to model the real-time non-deterministic processes in Uppaal in terms of state space using the built in clocks and communicating channels. This feature along with the availability of graphical user interface (GUI), communication channels, and counter example details were among the main motivations behind using the Uppaal model-checker for this work.

3. Proposed Methodology

The comprehensive proposed methodology for the timing analysis of digital circuits is illustrated in Figure 2. The proposed methodology has four primary steps: delay calculation, path extraction, path modeling and property verification in Uppaal.

3.1. Delay Calculation

The first step in the proposed methodology is to compute the resistance and capacitance values for $NMOS$ and $PMOS$ transistors in the ON state using the basic technology parameters. This way, the timing models are proposed for the basic circuit components, i.e., NAND, NOR, NOT and a Flip-Flop. Complex circuits are further modeled with the help of these basic gates. $NMOS$ and $PMOS$ [36] gate capacitances are given below:

$$C_{gatenMOS} = C_{gminN} \times fan-out \times WR_{nMOS} \tag{6}$$

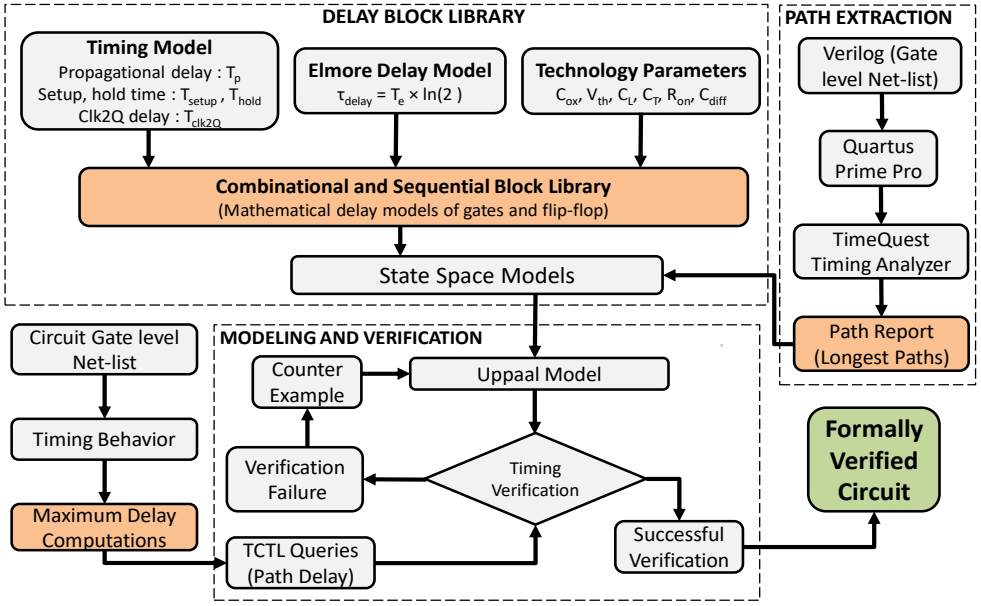


Figure 2: Block Diagram of a Proposed Methodology

$$C_{gatepMOS} = C_{gminP} \times fan-out \times WR_{pMOS} \quad (7)$$

Where C_{gmin} shows the minimum value of the gate capacitance and WR depicts the width ratio. C_L is the load capacitance which is computed based on sum of gate capacitances of all the gates attached at the output of the considered component.

$$C_L = \sum_{k=1}^{a} C_{gatenMOSk} + \sum_{j=1}^{b} C_{gatepMOSj} \quad (8)$$

The diffusion capacitance C_{Diff} can be computed with the help of drain capacitance [45]. The sum of load and diffusion capacitance allows us to obtain the gate's total capacitance C_T, which is further used for the calculation of gate delays.

$$C_T = C_L + C_{Diff} \quad (9)$$

The resistance of a $PMOS$ or $NMOS$ [37] can be calculated using the below formula:

$$R_{on} = \frac{1}{WL \times \mu \times Cox \times (V_{GS} - V_{TH})} \quad (10)$$

Using the values of corresponding capacitances and resistances, the Elmore delays can be computed for the NOT, NOR and NAND gates. Delay is computed on the basis of all the possible input transitions of a particular gate. For example, the transistor level diagram of the NAND gate and the corresponding equivalent circuit in terms of internal resistances (R_p and R_n) and capacitance (C_T and C_{ST}) is shown in Figure 3. In this RC model of the NAND gate, the resistors act like a switch. The switching of resistors depend upon the applied input voltages. For example, if a 0

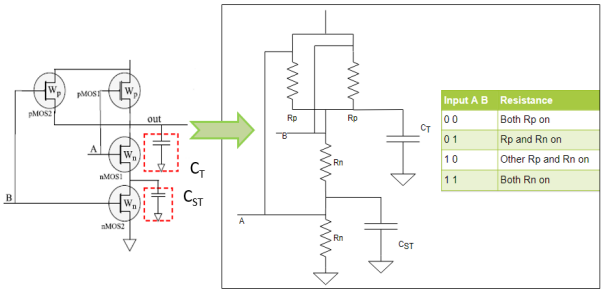


Figure 3: Transistor Level Diagram and Equivalent RC Model of NAND Gate

input is applied at both terminals A and B then both PMOS resistances act as short circuits, i.e., no resistance, while both NMOS resistances act as open circuits or a resistance of infinity. Similarly, for inputs of 0 and 1, the PMOS resistor connected with terminal A, and NMOS resistor connected with terminal B acts as a short while the other resistors act as an open circuit. In the same way, with 1 and 0 inputs, the resistor switching is opposite to the previous case while for both 1s as an input, both NMOS resistors act as short circuits. These switching behaviors help us to find out the equivalent resistances and capacitances, and therefore allows us to derive the Elmore delay equations for the NAND gate. In a similar way, we build the RC models of NOT and NOR gates, analyze the resistor switching and derive the corresponding Elmore delay equations for the NAND, NOR and NOT gates as shown in Tables 1, 2, and 3,.

True Single-Phase Clocked (TSPC) Flip-Flop model is used in our model [36] for analyzing the timing characteristics of a Flip-Flop because of its less complexity and minimum number of transistors to deal with [36]. As discussed earlier, clock to Q delay, setup time, and the

Table 1
Delay Equations of a Nand Gate

Input	Output	Delay Equation
00	1	$ln(2) \times [(C_T \times R_p)/(2 \times WR_{pMOS})]$
01	1	$ln(2) \times [(C_T \times R_p)/WR_{pMOS}]$
10	1	$ln(2) \times [((C_T + C_{ST}) \times R_p)/WR_{pMOS}]$
11	0	$ln(2) \times [(C_T \times 2 \times R_n)/WR_{nMOS}]$

Table 2
Delay Equations of a Nor Gate

Input	Output	Delay Equation
00	1	$ln2 \times [2 \times (C_T \times R_p)]/(WR_{pMOS})$
01	0	$ln2 \times [(C_T \times R_n)/WR_{nMOS}]$
10	0	$ln2 \times [((C_T + C_{ST}) \times R_n)/WR_{nMOS}]$
11	0	$ln2 \times [(C_T \times R_n)/(2 \times WR_{nMOS})]$

Table 3
Delay Equations of a Not Gate

Input	Output	Delay Equation
0	1	$ln2 \times [(C_T \times R_p)/WR_{pMOS})]$
1	0	$ln2 \times [(C_T \times R_n)/WR_{nMOS})]$

Table 4
Delay Equations of a Flip-Flop

Input	Output	Delay Equation
T_{setup}		
0	0	$ln(2) \times [(C_T \times R_p)/WR_{pMOS})]$
1	1	$ln(2) \times [(C_T \times R_n)/WR_{nMOS})]$
T_{hold}		
0	0	$ln(2) \times [(C_T \times R_p)/WR_{pMOS})]$
1	1	$ln(2) \times [(C_T \times R_n)/WR_{nMOS})]$
T_{Clk2Q}		
0	0	$ln(2) \times [(3 \times C_T \times R_p)/WR_{pMOS})]$
1	1	$ln(2) \times [(3 \times C_T \times R_n)/WR_{nMOS})]$

hold time, are three important constraints in a Flip-Flop. In the TSPC Flip-Flop model, setup time is considered to be equivalent to a single inverter delay, the hold time is considered to be less than an inverter delay and thus three times inverter delay becomes the propagational delay. Similarly, during modeling in Uppaal, we consider the hold time to be equal to one inverter delay in the worst case. The delay equations presented in proposed model for the clock to Q delay, setup time, and the hold time are provided in Table 4.

3.2. Path Extraction

The propagation delay of a digital circuit, which is composed of numerous logic gates and Flip-Flops, can be calculated by considering all of its paths, i.e., the paths from a primary input to the input of a Flip-Flop, between Flip-Flops, and from the output of a Flip-Flop to a primary output. The path delay can be computed by adding the delays of all the logic elements present in that path. We can manually identify and analyze all timing paths present in a circuit in case of smaller circuits and can calculate the delays of all the paths. However, in case of bigger circuits, with thousands of gates, it is practically impossible to identify and analyze all the paths manually. Therefore, we propose to use Altera Quartus Prime Pro [35] to automatically identify valid paths of a circuit, expressed in a hardware description language (HDL). It provides the detailed path information for both combinational and sequential circuits. In case of combinational circuits, it provides all the possible paths from all input ports to all output ports. During path analysis of sequential circuits, it provides paths from the input port to a Flip-Flop, Flip-Flop to a Flip-Flop, and Flip-Flop to an output port.

For the path extraction step of the proposed methodology, we mainly require the Verilog [43] code of the circuit that needs to be analyzed, as it is a widely used hardware description language (HDL). We compile the code and make sure that there is no syntax or logical error. After compilation, we synthesize the code and run the TimeQuest Timing Analyzer tool to obtain the paths' information of a given circuit based on the Synopsys design constraint file (SDC).

3.3. Path Modeling

In order to perform timing verification of digital circuits, modeling and verification using the Uppaal model-checker is the most important step in the proposed technique. In this process, the first step is the discretization of the delay using integers for delay modeling in Uppaal. The actual delay values of basic gates, i.e., NAND, NOR, and NOT gates are floating point exponentials in the order of picoseconds. However, Uppaal does not support floating point exponentials. Therefore, we have to discretize the delays in such a way that they can be analyzed by the Uppaal model-checker. We divide the overall delay value into the integral and exponential part. We perform Uppaal modeling and verification using the integral part and combine the exponential part with the integral part at the end after the verification in Uppaal. The discretization process of delays in Uppaal is depicted in Figure 4.

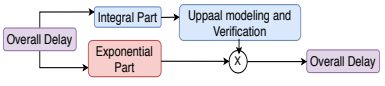


Figure 4: Discretization of Delay in Uppaal

In the proposed methodology, instead of modeling the complete circuit for finding out the maximum delay, we propose to use the longest path information for delay modeling. Information about the logic elements present in the longest path can be obtained using the Quartus Prime Pro Timing

Analyzer. The longest paths are translated into the state space and then modeled in Uppaal model-checker in terms of basic gates i.e., NAND, NOR, Not. In this way, we can find out the timing parameters i.e., maximum delays, clock time period, e.t.c., with a significantly reduced state space size. We primarily check that the longest path delay is less than or equal to the required maximum delay. If the delay exceeds the maximum value, then the Uppaal model-checker provides the counterexample along with the exact trace that gives information about the timing violation. Thereafter, in this way we can find out the real cause of property violation that whether it is due to an actual timing violation or a modeling error.

In order to facilitate the modeling of digital circuits, we perform the formal modeling of basic gates, i.e., NAND, NOR, NOT and a Flip-Flop using Uppaal model-checker and then these basic models are built upon to develop formal models of bigger complex circuits. The generic Uppaal model of basic logic gates is shown in Figure 5 for delay calculations. A binary value of 0, or 1, is assigned to the input variable in the initial state $S1$. Total resistances and capacitances of a gate is calculated using various technology parameters in the second state $S2$. Elmore delay is updated from the overall resistances and capacitances of a logic gate. Inputs values are acting as a guard in the third state $S3$. A guard is a side-effect free function that returns a boolean expression. Depending on the input value, logic gate delay is calculated. For a single input gate, e.g., NOT gate, we have two guards and based on the guard value, the delay is calculated e.g., when guard (input) is 0, PMOS is ON and NMOS is OFF and therefore delay will be the product of C_T and R_P. Similarly when guard (input) is 1, only NMOS is ON and the delay equation will be the product of C_T and R_N. For the two input universal gates, four different combinations of guards are used for delay calculations as shown in Figure 5. Finally, the output is updated depending upon the type of logic gate in state $S4$. After the state $S4$, all the delays and output values are reset and the cycle repeats. It is to be noted that Figure 5 is an abstract level diagram giving generic overview about modeling of basic gates using state transition diagram, therefore committed states are not shown here. Committed states are displayed only in exact Uppaal model (e.g., Figure 7). In the proposed technique, delay is an integer variable that updates on a transition depending upon the given input combination therefore states are made committed to avoid any delay in states.

The timed automata of the NOT gate in Uppaal model-checker is shown in Figure 7a where input is *xin_not* and output is represented by *xout_not*. Boolean value 0 or 1 is assigned to the input *xin_not* by the selection expression *xin_not: int[0, 1]* in the initial state. The fan-out *fo_not* is computed by noticing the total number of load gates attached at the driving gate output port. Elmore delay equation is used for the calculation of NOT gate delay *delay_not* on the basis of the inputs, internal capacitances, internal resistances, fan-out, and numerous technology parameters. Once the gate delay has passed, the gate output gets updated.

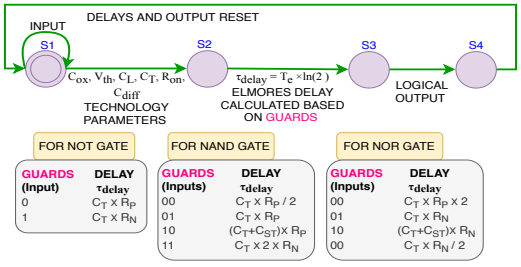


Figure 5: Generic Logic Gates Model

The gate output gets its appropriate value, i.e., the negation of input *out_not:= !(xin_not)*, after the delay has elapsed. In the similar way, the models of NOR and NAND gate have also been developed. Figure 7 b depicts the timed automata of the NAND gate in Uppaal. These basic gates can be used to formalize any combinational gate-level circuit. Sequential circuits contain Flip-Flops along with the logic

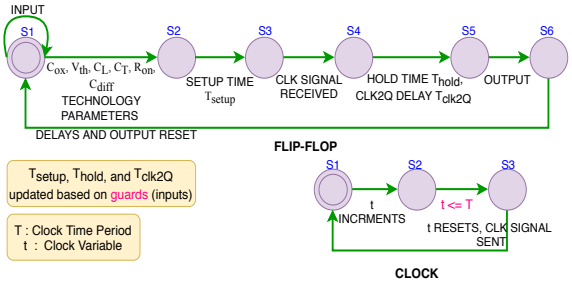


Figure 6: Generic Flip-Flop and a Clock Model

gates. The generic model of a proposed Flip-Flop along with the clock signal as illustrated in the Figure 6. The clock model has three states. In the first state $S1$, clock variable t increases until it reaches the clock time period T. In state $S3$, clock cycle completes and the clock variable t resets. As soon as the clock cycle completes, clock signal is sent to the Flip-Flop. The input signal is updated in the first state $S1$ in the Flip-Flop model. On the basis of the inputs, internal capacitances, internal resistances, and numerous technology parameters, i.e., the setup time, hold time and clock to Q delays are computed using the equations of Elmore delay. Similar to the gate model, input values act as a guard in States $S3$ and $S5$. The setup timeT_{setup}, hold timeT_{hold} and clock to Q delaysT_{Clk2Q} are calculated depending upon the guard values. In State $S6$, the input value is assigned to the output after the clock cycle and then the output and delays are reset.

In the proposed approach, we can build models of larger and more complex circuits by integrating the basic logic gate and Flip-Flop models. In this way, we have developed a library of gate models and all the circuits can be implemented using these models. The AND gate up to 4 inputs, 3 inputs OR gate, and a sequential circuit's path are shown in Figure 8. Similarly, NOR, OR and other higher input gates are implemented.

(a) NOT Gate (b) NAND Gate

Figure 7: Timed Automata (TA) of NOT and NAND Gate

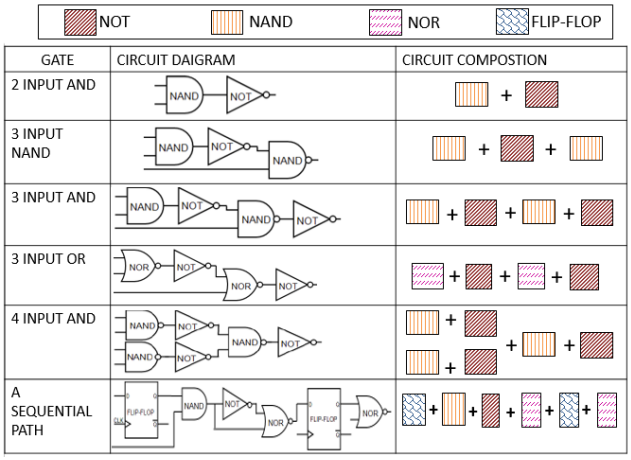


Figure 8: Use of NAND, Not, NOR Gates for Higher Inputs Gates' Circuits

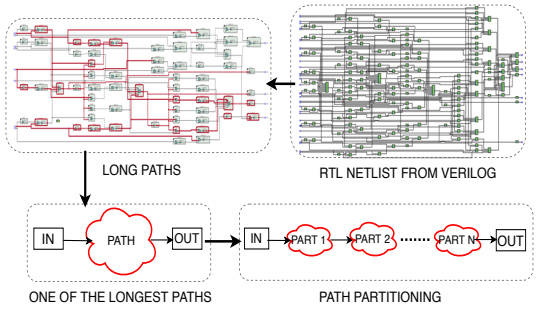


Figure 9: Generic Path Partitioning Technique

We mainly propose to model the longest paths in Uppaal instead of modeling the complete circuit to overcome the state space explosion issues. In a combinational circuit, delay of the longest path is measured directly from input port to output port. However, in a sequential circuit, the longest path is identified by observing three kinds of paths, i.e., from the input ports to the inputs of the Flip-Flop, between Flip-Flops, and from Flip-Flops to the output ports. We mainly focus on clock time period in case of sequential circuits. While analyzing the delays in a combinational or sequential circuit, there could be very large paths that can lead to state space explosion problem as well. In such cases, we propose to partition such path into smaller parts as shown in Figure 9. The main idea is to split the longer paths, that we obtain from RTL netlist, in such a way that the total delay of the original path is equal to the sum of the delay of the smaller parts. Consider the longest path P with delay D_T. The first step is to divide the path P into N parts. D_{P1}, D_{P2}, and D_{PN} are delays of the partitioned paths $part1$, $part2$, and $partN$, respectively. Since we divide P into N parts, we can say that the total delay D_T is equal to the sum of delays of all the parts of a considered path.

$$D_T = D_{P1} + D_{P2} \cdots \cdots + D_{PN} \tag{11}$$

$D_{\max}$ is the maximum possible delay of a path. With the help of the Uppaal model-checker, we will verify that the sum of delays of all the parts of a considered path cannot exceed $D_{\max}$ as stated in the following equation,

$$D_{P1} + D_{P2} \cdots \cdots + D_{PN} \leq D_{\max} \tag{12}$$

From Equations 11 and 12, we can imply

$$D_T \leq D_{\max} \tag{13}$$

The proposed path partitioning is illustrated in Figure 10 in which one of the paths of the C17 benchmark circuit is modeled and then divided into 3 parts. Each part is modeled as a separate timed automata and verified with a separate property. The overall verification time is obtained by adding the time of each part and the overall memory utilization is considered to be the maximum of memories utilized by the three parts as illustrated in the equations below where T_T represents the total verification time and M_T represents the overall memory utilization during verification by a circuit. The effect of path partitioning and its comparison with STA approach is shown in Section 4.3.

$$T_T = t_1 + t_2 + \cdots \cdots + t_N \tag{14}$$

$$M_T = \max(m_1, m_2, \cdots \cdots, m_N) \tag{15}$$

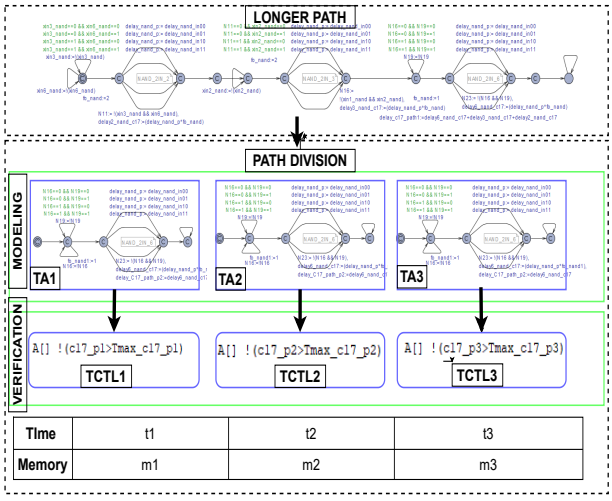


Figure 10: Time and Memory Management for a Longer Path in Uppaal model-checker for C17 Benchmark Circuit

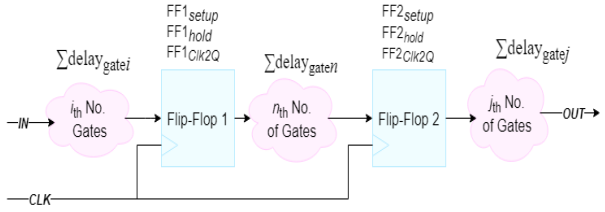


Figure 11: A Typical Sequential Circuit

3.4. Property Verification

Now we will discuss the TCTL properties for analyzing timing behavior of circuits, like maximum delay, time period of a clock, setup and hold time constraints e.t.c.,

1. In order to ensure that timed automaton is not stuck in any specific state, we mainly check the deadlock property. *Deadlock* is a predefined predicate in Uppaal.

$$\forall \square \ not \ \ deadlock$$

2. For the verification of combinational circuits, we check the delay of the given path is less than the allowed margin, i.e., the maximum allowed value. We receive a counterexample when circuit delay exceeds the maximum value and property verification fails. If the delay exceeds the maximum value and the property fails then we get a counterexample. Counterexample can be very useful in tracing the exact cause of property failure.

$$\forall \square \ !((delay_{g1} + delay_{g2} + \cdots \cdots + delay_{gn}) > D\max_{comb}))$$

Where $delay_{g1}$ shows the delay of a first gate, and $D\max_{comb}$ represents the maximum delay allowed as per the timing requirements of the given circuit.

3. For the verification of sequential circuits, we examine the path delays from input ports to Flip-Flops and then from Flip-Flops to output ports, in a similar way as we analyze the propagational delays of combinational circuits. Furthermore, for analyzing the Flip-Flop to Flip-Flop path, we consider the T_{setup} and T_{hold} time constraints to find out the clock period T of a given sequential circuit and to prevent metastability. For example, if we consider a typical sequential path, shown in Figure 11, where IN is an input port, $FF1$ and $FF2$ are two Flip-Flops, and OUT is an output port. We consider i gates between the input port IN and a first Flip-Flop $FF1$, n gates between the two Flip-Flops, and j gates between a second Flip-Flop $FF2$ and an output port. Following properties are proposed to be verified in this case.

$$\forall \square ((delay_{g1}+delay_{g2}+\cdots+delay_{gi}) \leq D\max_{INtoFF})$$

which states that the sum of delays of all the logic elements from input port IN to the Flip-Flop $FF1$ must be less than or equal to $D\max_{INtoFF}$, where $D\max_{INtoFF}$ is the maximum delay value that a path from input port to a Flip-Flop can attain.

$$\forall \square (T \geq (FF1_{clk2Q}+delay_{g1}+delay_{g2}+\cdots+ delay_{gn})+FF2_{setup})$$

This property avoids setup time violation by adding the clock to Q delay of first Flip-Flop $FF1$, delays of all the logic elements between Flip-Flops and setup time of second Flip-Flop $FF2$. This property determines the minimum required time period of clock T.

$$\forall \square ((FF1_{clk2Q}+delay_{g1}+delay_{g2}+\cdots+ delay_{gn}) \geq FF2_{hold})$$

This property avoids hold time violation by ensuring that the clock to Q delay of first Flip-Flop $FF1$ and all the logic elements between Flip-Flops must be greater than or equal to the hold time of second Flip-Flop $FF2$.

$$\forall \square ((FF2_{clk2Q}+delay_{g1}+delay_{g2}+\cdots+ delay_{gj}) \leq D\max_{FFtoOUT})$$

This property determines the maximum delay from the second Flip-Flop $FF2$ to the output port OUT by adding the delay of second Flip-Flop $FF2$ and all the logic elements from $FF2$ to the output port OUT.

We propose Algorithm 1 for identifying the maximum delay of a path, and a required time period of a clock to avoid the setup and hold time violations. This algorithm computes the delay $Temp\_Delay$ based on the analysis of counter examples. We mainly check if the total delay of a path is greater than $Temp\_Delay$. If yes, Uppaal will generate a counter example showing the delay value greater than $Temp\_Delay$,

and then $Temp\_Delay$ value is updated with a new delay value by adding the difference of delay value in it. This process continues until $Temp\_Delay$ becomes the maximum delay value and the property gets satisfied as shown in Lines 13 of an Algorithm 1. After this, $Temp\_Delay$ value is saved in $Tmax\_Delay$, which represents the maximum delay of a path. Process outlined in Algorithm 1 is done manually. However, it can be automated using a simple script as the process involves repeatedly invoking the UPPAAL model checker until a property corresponding to the delay of the circuit is satisfied. Once the circuit is formally modeled, it is formally verified against the given property according to the Algorithm 1 to obtain the maximum delay of a circuit. The modeling of the circuit using state transition diagram and property writing is completely manual while property verification and generation of a counter example is fully automatic.

Algorithm 1 Maximum Delay Computations

1: *Tmax_Delay*: Maximum delay of a path
2: *Delay*: Circuit path delay value computed by Uppaal at run time
3: *Temp_Delay*: Temporary delay value given at instance in property.
4: *Time_Clock*: Time period of a clock in a particular sequential circuit
5: *Delay_FF*: Delay of a path present between two Flip-Flops computed by Uppaal at run time
6: *Temp_Clock*: Temporary clock time period given at instance in property.
7: *Analysis*:
8:
9: **do** ▷ Finding maximum delay of a path
10: Generate counter example
11: Temp_Delay ← Temp_Delay + δ ▷ δ is a time difference computed by analyzing counter example.
12: **while** (Delay > Temp_Delay);
13: Tmax_Delay ← Temp_Delay ▷ Store the maximum delay value
14:
15: **do** ▷ Finding maximum time period of a clock
16: Generate counter example
17: Temp_Clock ← Temp_Clock + δ
18: **while** (Delay_FF > Temp_Clock);
19: Time_Clock ← Temp_Clock ▷ Store the maximum delay value in clock time period

4. Case Studies

Using the proposed approach, timing properties of C17, and S27 benchmark circuits are formally verified for illustration purpose.

4.1. C17 Benchmark Circuit

C17, as shown in Figure 12(a), is one of the benchmarks from ISCAS-85. It consists of 6 NAND gates having 5 input ports and 2 output ports. One of the longest path of the C17 circuit generated from Quartus Prime Pro Timing Analyzer is modeled in Uppaal as shown in Figure 13. This path consists of NAND2, NAND3, and NAND6 gates. Each of these three gates have two inputs. First two gates have a fan-out value of 2. This path is verified in 0.01sec utilizing 9.6 MB memory. During the complete path modeling as shown in Figure 13, since N2, N3, and N6 are direct external inputs to NAND2 and NAND3 gates, therefore non-deterministic modeling of these inputs will not result in a pessimistic delay. NAND6 gate does not have any external input, instead its inputs are basically outputs of the NAND3 and NAND4 gates. Unlike NAND3, NAND4 gate is not a part of the considered longest path, and only its output affects the delay of the NAND6 gate indirectly. So, instead of completely modeling the NAND4 gate and providing its logical output to NAND6 gate, we non-deterministically model NAND4's output (NAND6 second input), and compute the worst case delay of a path by considering delay against both 0 and 1 logical inputs. In this way, some pessimism can be introduced in the complete path analysis due to off-path logic that affects the path delay. The property that is verified against the modeled path is as follows, where $Tmax$ shows the largest allowable delay value for a specific path i.e., 17.72 ps

$$\forall \square \ (!(c17_{p1} > Tmax_{c17-p1}))$$

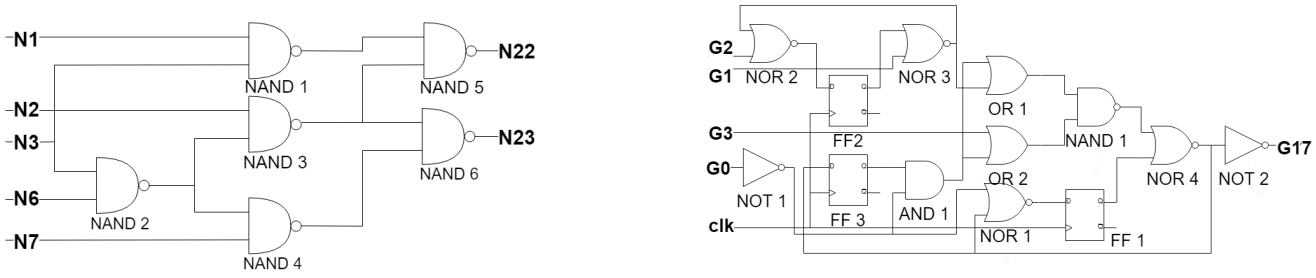
Other paths of C17 benchmark circuit have also been verified and details can be found in [6].

In order to illustrate the working of the proposed path partitioning technique, we partitioned the path of Figure 13 into two parts, as shown in Figure 14. We modeled these two parts in separate Uppaal templates and verified with specific properties in order to obtain the delay of each part. We split the path before NAND6 gate. Hence, the delay of NAND6 will not be dependent on the output of NAND3 for delay computations as the NAND6 gate is in a separate sub-path, so its delay will be computed for any of its four input vectors. Contrary to the above-mentioned complete path analysis, here both the inputs of NAND6 gate are modeled non-deterministically considering both of them as off-path logic, and thus resulting in some pessimism in an overall delay of a path. By modeling the off-path logic and side inputs non-deterministically, delay will be a bit overestimated. Due to path partitioning between NAND3 and NAND6, both parts are modeled as independent templates. The total delay of this path can be calculated as the sum of delay of these two sub-paths as shown below where $Tmax_{C17-p1_T}$ is 17.80 ps. This property is verified in 0.001 sec, utilizing 6.5MB of memory.

$$\forall \square \ (!(c17_{p1_1} + c17_{p1_2} > Tmax_{c17-p1_T}))$$

4.2. S27 Benchmark Circuit

S27, as shown in Figure 12(b), is one of the sequential circuit benchmarks from ISCAS-89. It has 10 gates and 3



(a) C17 Circuit (b) S27 Circuit

Figure 12: ISCAS Benchmark Circuits

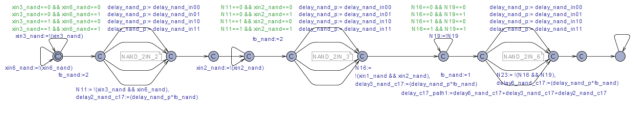


Figure 13: Timed Automata for C17

Flip-Flops with 4 input ports and 1 output port. S27 circuit is modeled in Uppaal considering the paths from input ports to Flip-Flops, between the Flip-Flops, and from Flip-Flops to output ports. The paths are obtained from the Quartus Prime Pro Timing Analyzer's path report. However, due to their relatively larger size of the state space, we did not represent the timed automata here. The details of the modeled path in terms of gates are as follows:

```
delay__p1_in=( delay2__or+delay1__nand+delay4__nor); //
    G3 --> FF3

delay__p1_ff=(delay1__clk2Q+delay4__nor+delay3__setup);
    // FF1 --> FF3

delay__p1_out=(delay2__clk2Q+delay3__nor+delay1__or+
    delay1__nand+delay4__nor+delay2__not); // FF2 -->
    G17
```

Properties that has been verified for the above mentioned paths are shown below. Other properties of this benchmark circuit for various other paths has also been verified and one can see the detail in [6]. $Tmax$, shows the largest value of delay and T_{clk}, depicts the clock time period in these properties.

$\forall \square (delay_{p1-in} \leq Tmax_{p1-in})$
$\forall \square (T_{clk} \geq (delay_{p1-ff}))$
$\forall \square (delay_{p-h1} \geq (delay_{3-hold}))$
$\forall \square (delay_{p1-out} \leq (Tmax_{p1-out}))$

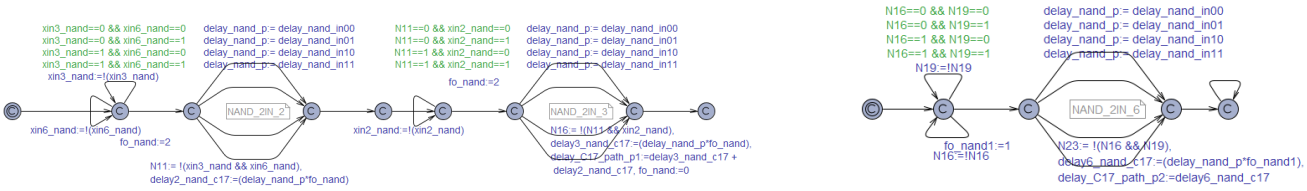
4.3. Results of Timing Verification

All the case studies are executed on 2.10 GHz 64-bit Window with 6GB RAM. It is important to note that traditional STA does not consider the circuit functionality while calculating the delays but the proposed technique allows us to compute the delays while considering all input combinations and the circuits' functionality along with delays of each component.

Minimum and maximum delays of NAND, NOR, AND, OR logic gates with multiple inputs and their comparison with STA is shown in Figure 15. A comparison of results from the proposed path partitioning technique with STA approach is shown in Table 5 and Figure 16 with some ISCAS'85 case studies. Due to path partitioning, although delays are little bit deviating from original values but far more realistic than the STA approach. The main justification for this is that in the proposed technique, we are considering all input vectors and the functionalities the of circuits. We can also observe that we have slightly compromised the accuracy of the results to make the approach more scalable as the results are not as accurate as the ones obtained without partitioning but they are much more realistic than STA.Due to the modeling of side inputs and off-path logic non-deterministically in the partitioned paths, the reported delay of the overall circuit is slightly overestimated in the proposed approach. The differences of delays between the analysis of the complete path and the partitioned path is shown in Table 5. Based on the considered cases, it can be easily observed that the difference is less than 1%. Moreover, the greater the the off-path logic and the number of side inputs, the greater the difference is, but even with the error the reported delays are more accurate than the results obtained through traditional STA. Comparison of clock time period of sequential circuits is shown in Table 6.

The considered combinational circuits and their verification statistics while verifying the longest paths in a circuit are summarized in Table 7 using the information about the total number of gates, its verification time, and the overall memory utilization during the verification phase of the corresponding circuit's path. Modeling and verification details of longest paths of sequential circuits are summarized in Table 8, providing the information about the total number of gates and Flip-Flops, verification time, and the overall memory consumption by state space during the verification process.

We have noticed significant increase in the number of explored states during verification with an increase in the state space size. For example, the total number of explored states in the C6288 benchmark circuit is 311272 whereas 130528 states were explored while analyzing the S5378 benchmark circuit. These explored states are much less in



(a) (b)

Figure 14: Timed Automate of C17 Partitioned Path.

Table 5
Comparison of Delays of Complete Path, Partitioned Path, STA Approach

Circuits	Comp Path (ps)	Part Path(ps)	Diff(ps)	STA (ps)
C17[17]	17.72	17.80	0.08	17.80
C432[17]	145.70	145.70	0	186.36
C499[17]	172.69	173.91	1.22	225.32
C1908[17]	115.74	116.66	0.92	120.32
C3540[17]	Can't model	468.31	-	520.46
C6288[17]	Can't model	663.78	-	689.76

Table 6
Comparison of Clock Time Period with STA approach

Circuits	Path Delays (ps)	STA Delays (ps)
S27 [16]	27.24	32.62
S208 [16]	122.39	131.48
S386 [16]	85.82	104.48
S820 [16]	419.90	427.8
S1423 [16]	156.48	169.80
S5378 [16]	84.59	111.64

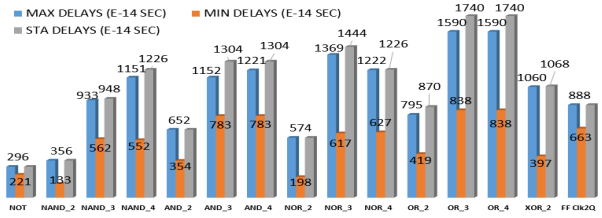


Figure 15: Comparison of Logic Gate Delays of Proposed and STA Technique

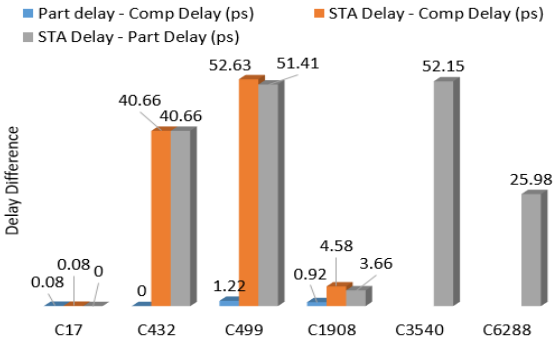


Figure 16: Comparison of Delay Difference using Three Different Approaches

number in comparison with [7], where the complete circuit is modeled in Uppaal instead of the paths.

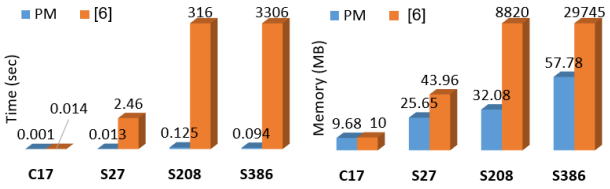
In comparison with an existing approach, presented in [7], where Uppaal model-checker is used for modeling and verification of digital circuits, we find the proposed technique to be much more efficient as shown in Figure 17(a). These results are based on the analysis of maximum time and memory consumption by the Uppaal model-checker. For example, in case of the C17 circuit, the verification time in [7] and the proposed technique is 0.014 s and 0.001 s, respectively. In [7], the complete circuit is modeled in Uppaal for timing verification while only the longest path, obtained from Quartus Prime Pro Timing Analyzer, is modeled in Uppaal in the proposed technique to cater for the state-space explosion problem. Comparison between the two approaches in terms of memory utilization is shown in Figure 17b. Scalability of the proposed approach can be observed from Figure 18 where the runtime is increasing with larger number of gates.

Table 7
Time and Memory Utilization in Combinational Circuits

Comb circuits	Max. Gates	Verification	
		Time(s)	Memory(MB)
C17 [17]	6	0.001	6.5
C432 [17]	160	1.985	340.2
C499 [17]	202	0.5	125
C1908 [17]	880	1.18	246.4
C3540 [17]	1669	17.743	1261
C6288 [17]	2416	44.90	3844.4

Table 8
Time and Memory Utilization in Sequential Circuits

Circuit	Verification		Max. Gates	FF
	Time(s)	Memory(MB)		
S27 [16]	0.013	50.83	10	3
S208 [16]	0.125	32.08	96	8
S386 [16]	0.094	57.78	159	6
S820 [16]	0.782	105.4	289	5
S1423 [16]	0.14	30.5	657	74
S5378 [16]	0.031	21.9	2779	179



(a) Verification Time of [7] and the Proposed Technique (b) Memory Utilized by [7] and the Proposed Technique

Figure 17: Comparison with Existing Technique [7]

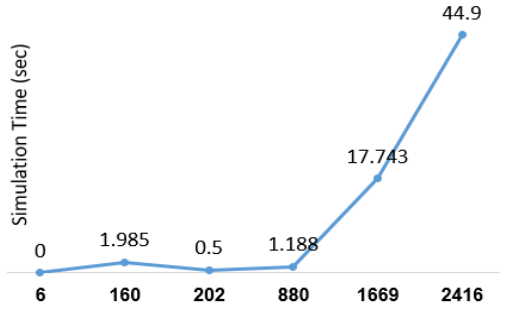


Figure 18: Comparison of Simulation Time and No. of Gates

A brief comparison of proposed technique with some existing approaches is represented in Table 9. This comparative analysis is primarily based on six parameters. The first two parameters depict the type of analyzed circuit, i.e, combinational or sequential circuit. The next two parameters show the techniques used for the delay modeling of circuits and the model-checker used for timing verification. At the end, the final two parameters refer to the maximum number of gates and Flip-Flops verified by the corresponding approaches. The proposed approach is proved to be better than already existing mentioned techniques in the following ways:

1. We proposed timing verification of both the combinational and sequential circuits unlike existing techniques [38], [2],[4],[3].

2. Contrary to assumed delay models as used in [13], [22], [38], [42], we determine the delay of logic gates using the Elmore delay modeling technique [2] to facilitate more realistic modeling and verification.

3. In comparison with all existing approaches of formal timing analysis, we model and verify digital circuits containing larger number of gates and Flip-Flops mainly because of the proposed path based modeling approach.

4. We proposed to compute the delays of digital circuits' by modeling and verification of longest paths using the Uppaal model-checker while considering the circuits functionality and all possible input vectors as well. We also proposed a path splitting technique in order to overcome the state-space explosion issue. Moreover, we presented an example to compare the delays computed through the proposed approach and the classical STA and our previously proposed unpartitioned formal functional timing analysis approach [7].

Table 9
Comparison of Proposed Technique with the Existing Approaches

Related Work	Comb circuits	Seq circuits	Delay Model	Tool	Max Gates	Max FF
Bozga et al. [13]	✓	✓	Assumed delay	Open-Kronos	24	4
Salah et al. [38]	✓	x	Assumed delay	Open-Kronos	88	x
Clariso et al. [22]	✓	✓	Symbolic delay	Abstract Algorithm	12	4
Bara et al. [10]	✓	✓	SPICE delay	Kronos/ Uppaal	100	15
Abbasi et al. [2]	✓	x	Elmore delay	nuXmv	68	x
Ain et al. [7]	✓	✓	Elmore delay	Uppaal	415	64
Abbasi et al. [4]	✓	x	Elmore delay	nuXmv	166	0
Abbasi et al. [3]	✓	x	Elmore delay	nuXmv	1669	0
Proposed Methodology	✓	✓	Elmore delay	Uppaal	2779	179

5. Conclusions

This paper presents a technique for formal timing analysis of digital circuits using a model checking based approach. Digital circuits are formally modeled in the form of state transitions diagrams and their timing behavior is formally verified using the TCTL queries with the help of the Uppaal model-checker Uppaal. Models of fundamental components of digital circuits, i.e., NAND, NOR, NOT gates and Flip-Fops have been developed and then these basic models are used in modeling of further bigger and complicated circuits. Hence, a generic framework has been developed to facilitate formal modeling of complex circuits upto 2779 gates. Our technique results in much reduced state space as we modeled the longest paths of circuits instead of modeling the complete circuit for timing analysis. The proposed approach provides automatic path extraction using Quartus Prime Pro that facilitates its usage for larger circuits. We also presented a path based partitioning technique to make analysis more scalable. An algorithm for the computation of maximum delays of circuits has also been proposed. This approach can be helpful in formal verification of several timing properties of digital circuits, i.e., determining the clock period of a circuit, finding out the setup time, hold time constraints and critical path in a circuit. For illustration purposes, we conducted the formal timing analysis of the ISCAS-85 and ISCAS-89 benchmark circuits. We can reduce the transitions by modeling only minimum and maximum delay values for each gates instead of modeling different delay values at each and every input combination. The reduced number of states and transitions will eventually require less memory for verification and thus makes the approach more scalable but would provide more pessimistic results. Future plan is to incorporate clock skew and routing delays in a circuit to obtain a more realistic and accurate timing model. The proposed approach can also be parallelized to be more scalable by utilizing many-core CPUs for multi-threaded applications.

References

[1] §, T., . A timed automaton-based method for accurate computation of circuit delay in the presence of cross-talk.

[2] Abbasi, I.H., Lodhi, F.K., Kamboh, A.M., Hasan, O., 2016. Formal verification of gate-level multiple side channel parameters to detect hardware trojans, in: Formal Techniques for Safety-Critical Systems, CCIS, vol. 694, Springer. pp. 75–92.

[3] Abbassi, I.H., Khalid, F., Hasan, O., Kamboh, A.M., 2019. Using gate-level side channel parameters for formally analyzing vulnerabilities in integrated circuits. Science of Computer Programming 171, 42–66.

[4] Abbassi, I.H., Khalid, F., Hasan, O., Kamboh, A.M., Shafique, M., 2018. Mcsevic: A model checking based framework for security vulnerability analysis of integrated circuits. IEEE Access 6, 32240–32257.

[5] Agarwal, A., Paul, B.C., Mukhopadhyay, S., Roy, K., 2005. Process variation in embedded memories: failure analysis and variation aware architecture. IEEE Journal of Solid-State Circuits 40, 1804–1814.

[6] ul Ain, Q., 2019. Formal Timing Analysis of Digital Circuits. Available online: http://save.seecs.nust.edu.pk/projects/ftadc/.

[7] Ain, Q.U., Hasan, O., 2018. Formal timing analysis of digital circuits, in: Formal Techniques for Safety-Critical Systems, CCIS, volume 1008, Springer. pp. 84–100.

[8] Alur, R., Courcoubetis, C., Dill, D., 1990. Model-checking for real-time systems, in: Logic in Computer Science., IEEE. pp. 414–425.

[9] Andraus, Z.S., Sakallah, K.A., 2004. Automatic abstraction and verification of verilog models, in: Proceedings. 41st Design Automation Conference,, pp. 218–223.

[10] Bara, A., Bazargan-Sabet, P., Chevallier, R., Ledu, D., Encrenaz, E., Renault, P., 2010. Formal verification of timed VHDL programs., in: Forum on Specification Design Languages., IET. pp. 80–85.

[11] Behrmann, G., David, A., Larsen, K.G., 2006. A tutorial on Uppaal 4.0.

[12] Bhasker, J., Chadha, R., 2009. Static timing analysis for nanometer designs: A practical approach. Springer Science & Business Media.

[13] Bozga, M., Jianmin, H., Maler, O., Yovine, S., 2002. Verification of asynchronous circuits using timed automata. Electronic Notes in Theoretical Computer Science 65, 47–59.

[14] Bozzelli, L., La Torre, S., 2009. Decision problems for lower/upper bound parametric timed automata. Formal Methods in System Design 35, 121.

[15] Braibant, T., 2011. Coquet: a coq library for verifying hardware, in: International Conference on Certified Programs and Proofs, LNCS, vol. 7086, Springer. pp. 330–345.

[16] Brglez, F., Bryan, D., Kozminski, K., 1989. Notes on the ISCAS'89 benchmark circuits. North-Carolina State University .

[17] Bryan, D., 1985. The ISCAS'85 benchmark circuits and netlist format. North Carolina State University 25.

[18] Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S., 2014. The nuXmv symbolic model checker, in: Computer Aided Verification, Springer. pp. 334–342.

[19] Celik, M., Pileggi, L., Pileggi, L., Odabasioglu, A., 2002. IC interconnect analysis. Springer Science & Business Media.

[20] Chevallier, R., Encrenaz-Tiphene, E., Fribourg, L., Xu, W., 2009. Timed verification of the generic architecture of a memory circuit using parametric timed automata. Formal Methods in System Design 34, 59–81.

[21] Chiu, C.H., Huang, T.W., 2022. Late breaking results: Efficient timing propagation with simultaneous structural and pipeline parallelisms .

[22] Clarisó, R., Cortadella, J., 2004. Verification of timed circuits with symbolic delays, in: Asia and South Pacific Design Automation Conference., IEEE. pp. 628–633.
[23] Gupta, R., Krauter, B., Tutuianu, B., Willis, J., Pileggi, L.T., 1995. The elmore delay as bound for rc trees with generalized input signals, in: Proceedings of the 32nd annual ACM/IEEE Design Automation Conference, pp. 364–369.
[24] Hasan, O., Tahar, S., 2015. Formal verification methods, in: Encyclopedia of Information Science and Technology, Third Edition. IGI Global, pp. 7162–7170.
[25] He, J., Guo, X., Meade, T., Dutta, R.G., Zhao, Y., Jin, Y., 2019. Soc interconnection protection through formal verification. Integration 64, 143–151.
[26] Henzinger, T.A., Ho, P.H., 1994. Hytech: The cornell hybrid technology tool, in: International Hybrid Systems Workshop, Springer. pp. 265–293.
[27] Huang, T.W., 2020. A general-purpose parallel and heterogeneous task programming system for vlsi cad, in: 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), IEEE. pp. 1–2.
[28] Huang, T.W., Guo, G., Lin, C.X., Wong, M.D., 2020. Opentimer v2: A new parallel incremental timing analysis engine. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 40, 776–789.
[29] Huang, T.W., Wong, M.D., 2015. Opentimer: A high-performance timing analysis tool, in: 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), IEEE. pp. 895–902.
[30] Irfan, A., Cimatti, A., Griggio, A., Roveri, M., Sebastiani, R., 2016. Verilog2SMV: A tool for word-level verification, in: Design Automation Test in Europe Conference Exhibition, pp. 1156–1159.
[31] Kilts, S., 2007. Static timing analysis. Advanced FPGA Design: Architecture, Implementation, and Optimization , 269–278.
[32] Lee, P.Y., Jiang, I.H.R., Li, C.R., Chiu, W.L., Yang, Y.M., 2015. itimerc 2.0: Fast incremental timing and cppr analysis, in: 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), IEEE. pp. 890–894.
[33] Nilsson, J., Evans, J., 2002. PSPICE manual using orcad release 9.2 for introductory circuits.
[34] Pérez Hernández, S., et al., 2020. Statistical model checking in approximate computing systems .
[35] Quartus Prime Standard Edition Handbook, 2015. Quartus prime standard edition handbook.
[36] Rabaey, J.M., Chandrakasan, A.P., Nikolic, B., 2002. Digital integrated circuits. volume 2. Prentice hall Englewood Cliffs.
[37] Razavi, B., 2002. Design of Analog CMOS Integrated Circuits. Tata McGraw-Hill Education.
[38] Salah, R.B., Bozga, M., Maler, O., 2003. On timing analysis of combinational circuits, in: International Conference on Formal Modeling and Analysis of Timed Systems., LNCS, vol. 2791, Springer. pp. 204–218.
[39] Saleh, R., Jou, S.J., Newton, A.R., 1994. Gate-level simulation, in: Mixed-Mode Simulation and Analog Multilevel Simulation. Springer, pp. 123–152.
[40] Shiraz, S., Hasan, O., 2018. A library for combinational circuit verification using the hol theorem prover. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37, 512–516.
[41] Strnadel, J., 2021. Using model checker to analyze and test digital circuits with regard to delay faults, in: 2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), IEEE. pp. 111–114.
[42] Takan, S., Guler, B., Ayav, T., 2015. Model checker-based delay fault testing of sequential circuits, in: Architecture of Computing Systems., pp. 1–7.
[43] Thomas, D., Moorby, P., 2008. The Verilog® Hardware Description Language. Springer Science & Business Media.
[44] VLSI Universe, 2013. Setup time and hold time basics. Available online: http://vlsiuniverse.blogspot.com/2013/06/setup-and-hold-basics-of-timing-analysis.html.
[45] Weste, N.H., Harris, D., 2015. CMOS VLSI design: A circuits and systems perspective. Pearson Education India.