Precise and Efficient Analysis of Combinational Logic Loops in Digital Circuits

Chenyun Yin
2200012818@stu.pku.edu.cn
Peking University
Beijing, China

Yang Xiao
2200012736@stu.pku.edu.cn
Peking University
Beijing, China

Yafei Li
2200012715@pku.edu.cn
Peking University
Beijing, China

ABSTRACT

Combinational logic loops can cause signal oscillations and multiple driving issues, affecting circuit correctness and power consumption. In this study, we propose an efficient method for analyzing these loops, based on the 2024 China Graduate "Chip" Design Contest problem. Using the VPI interface of Icarus Verilog, we automatically detect logic loops in gate-level netlists and analyze oscillation conditions. We introduce a minimal register insertion strategy to maintain functional correctness with minimal overhead. Our approach, supported by experimental results, efficiently detects and analyzes logic loops, especially in circuits with fewer than 10,000 gates.The implementation of our code is available at https://github.com/atarchive/PreciseLoopChecker.

CCS CONCEPTS

• **Hardware → Combinational circuits**; • **Theory of computation → Graph algorithms analysis**.

KEYWORDS

SAT Solvers, Circuit Simulation, Oscillation Detection, Multi-loop Analysis

1 INTRODUCTION

In modern digital circuit design, the presence of combinational logic loops can lead to a series of issues, including signal oscillations, unstable outputs, and increased power consumption.

This work accurately solves the following five problems formulated on large-scale combinational logic circuits:

Problem 1: Identify all strongly connected components in the circuit. Problem 2: Identify strongly connected components in the circuit where oscillations are impossible. Problem 3: For strongly connected components that may experience oscillations, find the external input conditions that trigger the highest number of gates causing the oscillation. Problem 4: Insert the minimum number of registers to break all strongly connected components that may experience oscillations. Problem 5: Modify the combinational logic circuit such that if the original circuit does not oscillate, the modified circuit's logic function remains unchanged; if the original circuit oscillates, the modified circuit does not oscillate, and the generated flag signal indicates oscillation.

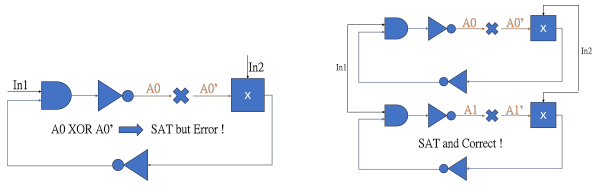
Note: The prerequisite for these five problems is that each strongly connected component contains at most three cycles.

The organization of the following content is as follows:

The first half of Section 3 introduces the three novel modeling and methods we proposed: "Multi-point SAT," "Child and Parent Propagation," and "Backward Propagation." The second half applies these methods to solve the above five problems. Section 4 analyzes the algorithm's complexity. Section 5 presents experimental results based on the existing test set. Section 6 concludes the work.

2 METHODOLOGY

2.1 Algorithm Approach



(a) Wrong Modeling (b) Corrct Modeling

Figure 1: 1-point SAT

2.1.1 Multi-point SAT. Multi-point SAT is a method used to analyze whether multiple points in a circuit can oscillate simultaneously using a SAT solver. First, to model whether the circuit can oscillate as a SAT problem, it is necessary to list the SAT expressions for circuit oscillation. For a signal on a wire, its oscillation behavior is characterized by the current value being exactly opposite to the value obtained through feedback in the circuit, regardless of whether the current value is 0 or 1. Therefore, the initial idea for determining whether a line can oscillate is to disconnect the line and explore whether a set of external inputs can be found such that the values at both ends of the disconnected line are opposite (XOR is 1). However, this is not enough, because the logic of feedback from 0 to 1 and from 1 to 0 must both hold simultaneously for the circuit to continue oscillating. Simply having the signals opposite does not guarantee that both conditions will hold. The solution is to duplicate the original circuit. The original circuit and the new circuit model the transitions from 0 to 1 and from 1 to 0, but they must share all external inputs, as they are essentially simulating two different circuit states of the same oscillating circuit within one oscillation cycle.

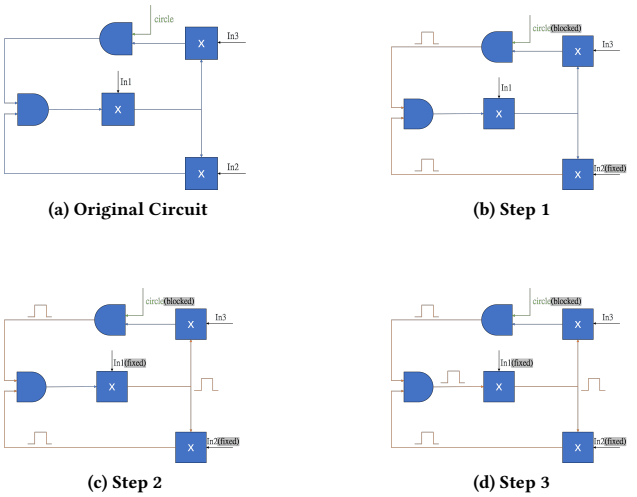
For the circuit in Figure 1 (b), if C(input, gates) represents the clauses that form the constraints of the circuit, then the SAT formula for oscillation detection at point A for this circuit is:

$$C(In1, In2, gates) \wedge C'(In1, In2, gates') \wedge (A0 \wedge \neg A0') \wedge (\neg A1 \wedge A1')$$

The method for 1-point SAT has been explained above. For m-point SAT, since the relationships between these points in the original circuit are unclear, when one breakpoint has values of 01, we

cannot determine whether the other points should be 01 or 10 (as the values of the external two-input gates are still unknown). Therefore, our strategy is to fix one point to 01 and then enumerate the remaining breakpoints as either 01 or 10 (still ensuring that the values of the original circuit and the complementary circuit are opposite). Finally, we take the disjunction of all the results to determine whether m points can oscillate simultaneously.

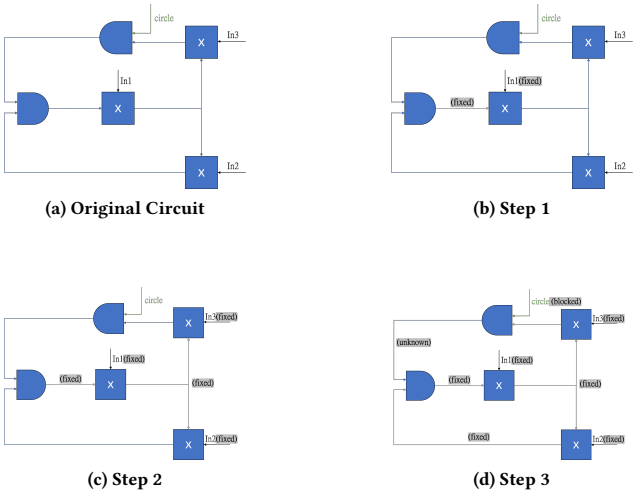
It is worth noting that in our later analysis, at most 3-point SAT is used, so the complexity of the "enumeration" operation is actually very low. Additionally, we utilized Tseitin encoding, which reduces the number of clauses by increasing the number of variables, making the solving process more efficient.



(a) Original Circuit (b) Step 1

(c) Step 2 (d) Step 3

Figure 2: Parent Propagation



(a) Original Circuit (b) Step 1

(c) Step 2 (d) Step 3

Figure 3: Child Propagation

2.1.2 Parent Propagation. The following conclusions can be made regarding the oscillating and stable states of the circuit:

- (1) If a signal oscillates, its parent node, which does not pass through a common two-input gate, also oscillates. In other words, the oscillating part of the circuit can propagate to the parent node (driving node) until reaching the output of a common two-input gate. This is because the external inputs of the circuit are fixed, and oscillation can only be caused by internal inputs of the parent node. The oscillation state of the two inputs of the common two-input gate is uncertain and cannot propagate;
- (2) The non-oscillating part of the circuit can propagate to the child node (driven point) until reaching an input of a common two-input gate. This is because the external input of the circuit is fixed, and the output cannot oscillate. The other input of the common two-input gate is uncertain, and thus, propagation is not possible.

The process of propagation in the child direction and the parent direction is described in detail in Figures 2 and 3.

The parent-directed propagation process in Figure 2 is as follows: The oscillating part is marked in orange. Assume that both inputs of the leftmost common two-input gate are oscillating. The oscillating part above will stop propagating due to another common two-input gate, while the part below will continue to propagate until the entire loop is evaluated. At this point, only a small portion of the circuit's state remains unknown.

The child-directed propagation process in Figure 3 is as follows: The non-oscillating part is marked in gray. As shown in the figure, the non-oscillating part propagates in the child direction. The upper part is blocked by a common two-input gate, while the lower part is not obstructed and continues to propagate.

2.1.3 Backward Propagation. For two-input gates such as AND gates, OR gates, NAND gates, and NOR gates, there is always an input value that makes the output independent of the other input. This value is called the deciding value, while the other value is referred to as the non-deciding value.

For example, 0 is the deciding value for AND and NAND gates; as long as one of the inputs is 0, the output will be 0 or 1, regardless of the other input.

If all parts of a circuit are oscillating, all of its external input values must be non-deciding values. If a circuit's starting node oscillates and the ending node does not, we propose a reverse propagation strategy to determine the external input conditions that maximize the number of oscillating gates.

Clearly, the transition from oscillating to non-oscillating must involve a two-input gate that "blocks" the oscillation through a deciding value from an external input. To maximize the number of oscillating gates, we need to minimize the distance between the blocking gate and the ending node. Thus, starting from the ending node, we reverse search for a blocking gate that can provide the current value.

For example, in Figure 4, A is a common two-input AND gate. One side is already determined to oscillate, and the other side must be the non-deciding value 1. Consider the part of the circuit from the output to the non-deciding input. The initial node oscillates, and the ending node is always 1. We start the search for the two-input gate that satisfies the blocking condition from the ending node. B is an AND gate; when the external input takes the deciding

value 0, the output is 0, so the external input of B remains a non-deciding value. C is a NOT gate; after passing through the NOT gate, the value changes from 1 to 0. D is an AND gate, and when the deciding value is 0, it provides an output of 0, so the external input of D is set to the non-deciding value. The part driven by D does not oscillate, while the circuit driving D must oscillate, with all external inputs set to the non-deciding value of the corresponding logic gate.

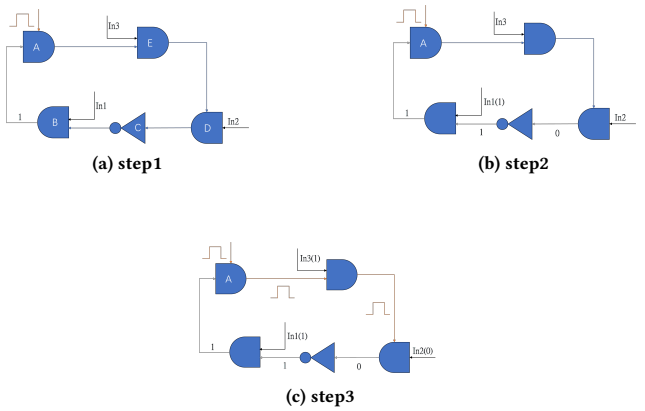


Figure 4: Backward Propagation

2.2 Application

2.2.1 Problem 1. Use the well-known Tarjan algorithm and Johnson algorithm to solve the strongly connected components and loops in combinational logic circuits.

2.2.2 Problem 2. For a single loop, it is sufficient to count the number of gates with "NOT" logic in the loop. For 2-loops and 3-loops, if the circuit oscillates, the common part of the loops will definitely oscillate. The output of the common two-input gate will be the common part. Therefore, starting from the output of the common two-input gate is the most efficient way to determine the oscillation. A 2-loop has only one common two-input gate, so we can use 1-point SAT to determine the oscillation. For a 3-loop, there are two common two-input gates. We first apply 2-point SAT to the outputs of both gates. If they cannot both oscillate, we perform 1-point SAT on each input. If the result is UNSAT, then the loop cannot oscillate.

2.2.3 Problem 3. **1-loop:** Based on the conclusions from Sections 2.1 and 2.3, all external inputs are non-deciding values.

2-loop: For solving the potentially oscillating circuits in Problem 2, to determine the oscillation state of each signal, we use the smallest child nodes, specifically the two input terminals of a common two-input gate, as breakpoints for SAT evaluation, since oscillation can propagate upward.

In the case of two loops, there is only one common two-input gate. Both input terminals are disconnected to perform a 2-point SAT evaluation. If the SAT solver returns **true**, it indicates that the entire circuit can oscillate simultaneously, and all external inputs should be set to non-deterministic values. Otherwise, it shows that only one loop in the circuit can oscillate. In this case, we disconnect only one of the input terminals, while the other remains connected, and perform a 1-point SAT evaluation. If the SAT solver returns **true**, then all external inputs in the oscillating part of the circuit should be set to non-deterministic values. Subsequently, backward search can be conducted from the non-oscillating input terminal.

3-loop: The solution for three loops is based on the approach for two loops. We need to select appropriate breakpoints to disconnect all loops that may oscillate before solving. In the case of three loops, there are four input terminals of common two-input gates. Using a mapping method, we can identify three input terminals that can disconnect all loops.

Next, these three points are simultaneously disconnected to perform a 3-point SAT evaluation. If the SAT solver returns **true**, it indicates that all three loops can oscillate simultaneously, and all external inputs should be set to non-deterministic values. Otherwise, only two or one of the loops oscillate, in which case the method for two loops is invoked to solve the problem.

2.2.4 Problem 4. Inserting a buffer is equivalent to breaking the oscillation loop. Applying the conclusions from Problem 2, inserting a buffer at the output of a common two-input gate is the most efficient method. For one-loop and two-loop cases, a single buffer is sufficient to break the loop. In the case of three loops, there are two output terminals, and we explore the possibility of breaking the loops with a single buffer.

Perform a single-point SAT evaluation on each output terminal individually. If both evaluations result in **UNSAT**, it indicates that two buffers are necessary to break the loops.

2.2.5 Problem 5. The solution is based on an event-driven simulator. The original oscillating circuit (e.g., the oscillating circuit shown in Figure 3) is modified into the circuit shown in Figure 5. An OR gate is inserted at one of the oscillating nodes, with the other input of the OR gate connected to a flag signal. The oscillating node is further connected to the clock input of a register. The D input of the register is set to a constant value of 1, and the Q output is initialized to 1'b0.

The Q output serves as the flag signal to indicate whether the circuit is oscillating. If oscillation occurs, a valid edge is generated at the clock input, causing the constant value of 1 at the D input to propagate to the Q output. As a result, the flag transitions from 0 to 1 and is fed back into the inserted OR gate, forcibly stopping the oscillation. This modification provides the simulator with a steady-state solution.

Figure 5: problem 5

3 COMPLEXITY ANALYSIS

3.1 Analysis for Problem 1 (Entire Circuit Analysis with V Nodes and E Edges)

- The Tarjan algorithm only requires one DFS on the graph, with a time complexity of $O(V+E)$, while the Johnson algorithm requires C DFS passes, resulting in a time complexity of $O(V+E)C$, which can be simplified to $O(V+E)(C+1)$. - In the problem, the maximum number of loops in each strongly connected component is 3, so the upper bound of the algorithm's time complexity is $O(4(V+E))$.

3.2 Analysis for Problems 2-4 (Analyze Each Strongly Connected Component, with n Gates and k External Inputs)

3.2.1 m-point SAT (Disconnecting m Edges and Using SAT to Model Oscillation Detection). **Number of Variables:** 1. The original circuit and the complementary circuit require $2n$ variables, while the external inputs remain unchanged, contributing k variables. Thus, there are $2n+k$ variables in total. 2. There are m breakpoints, adding m additional variables. 3. The values of the original circuit and complementary circuit on both sides of the breakpoints are combined using logical AND. The variables corresponding to the complementary circuit need to be negated. This results in $2m$ additional variables. 4. The product combinations to obtain the correct result give $2^{(m-1)}$ possible arrangements. Thus, $2^{(m-1)}$ additional variables are introduced (since only one arrangement is needed, the first breakpoint can be set arbitrarily). 5. The result variable contributes one additional variable.

In total, the number of variables is:

$$(2n + k + 3m + 2^{(m-1)} + 1)$$

Number of Clauses: 1. Constraints for the original circuit: Assuming the number of two-input gates and one-input gates is approximately the same, there are $2n \times (0.5 \times 2 + 0.5 \times 3) = 5n$ clauses. 2. Breakpoint product constraints: For each arrangement ($a0\&!a0\&!a1\&a1$), there are 5 clauses. There are $2m$ arrangements, contributing $10m$ clauses. 3. OR variable constraints: There are $2^{(m-1)}$ OR variables, and each variable's constraint involves combining m AND variables. This results in $(m+1) \times 2^{(m-1)}$ clauses. 4. OR variables combined into result variables: There are $2^{(m-1)}$ OR variables, and $2^{(m-1)} + 1$ clauses. 5. The result variable: 1 clause.

In total, the number of clauses is:

$$(5n + 10m + (m+2) \times 2^{(m-1)} + 2)$$

Let $S(n, m)$ represent the time complexity of solving a SAT problem with n variables and m clauses.

The SAT solver uses 1-point, 2-point, and 3-point SAT:
1-point SAT: $S(2n + k + 5, 5n + 15)$
2-point SAT: $S(2n + k + 9, 5n + 30)$
3-point SAT: $S(2n + k + 14, 5n + 52)$

3.2.2 Classification of Loops. **1-loop:**Only requires graph search, with a time complexity of $O(n)$. **2-loop:**For Problem 2, requires one 1-point SAT.For Problem 3, requires one 2-point SAT and two 1-point SATs. The complexity for reverse search and propagation is $O(n)$. **3-loop:**For Problem 2, requires at most one 2-point SAT and two 1-point SATs.For Problem 3, requires at most one 3-point SAT, three 2-point SATs, and three 1-point SATs. The complexity for reverse search and propagation is $O(n)$.

3.3 Analysis and Evaluation

We set a 1-second time limit for the SAT solver. The solver can handle around 20,000 variables and 100,000 clauses in this time. Assuming $k = n/2$ (i.e., the number of two-input gates and one-input gates in the strongly connected components are approximately equal), the estimated maximum values for n and k that can be solved within 1 second are approximately $n = 8000$ and $k = 4000$. This indicates that our algorithm is efficient and accurate for solving strongly connected components with up to around 10,000 gates.

4 EXPERIMENT

Our experiment is based on 16 existing test cases that already have answers. By comparing the results, we found that all answers are correct. The runtime and the graphs of each case are as follows:

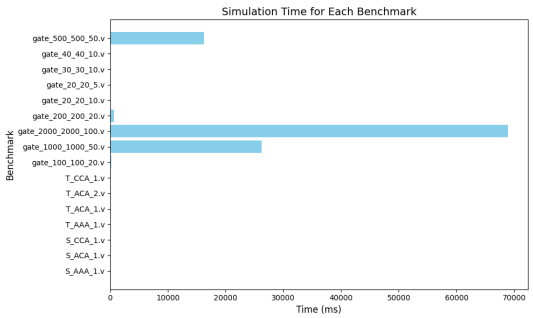


Figure 6: time for testcases

5 CONCLUSION

In this study, we focused on analyzing and solving oscillation issues in combinational logic circuits, particularly in strongly connected components (SCCs). Using algorithms like Tarjan and Johnson, we identified SCCs and loops, mainly focusing on loops with up to three nodes.

We developed SAT-based solutions to analyze oscillation conditions in various loop configurations, applying 1-point, 2-point, and 3-point SAT checks. Reverse search techniques further improved the accuracy and efficiency.

Our experiments on 9 test cases showed that our approach accurately identified oscillation conditions and provided solutions within expected runtimes, efficiently handling circuits with up to 8000 gates.

Overall, the methodology offers an efficient and reliable solution for analyzing oscillations in combinational circuits, suitable for circuit design and verification.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009