Efficient Simulation of Oscillatory Combinational Loops

Morteza Fayyazi
Mentor Graphics Corporation
Waltham, MA 02451, USA

Laurent Kirsch
Mentor Graphics Corporation
Waltham, MA 02451, USA

Abstract

This paper presents an efficient algorithm for post-synthesis logic simulation of digital circuits with oscillatory combinational loops. Oscillatory combinational loops can significantly degrade the performance of cycle accurate logic simulators. We provide an algorithm that first, dynamically detects oscillatory loops. Then, we introduce a novel approach to compute a multiple of their oscillation period which is used to optimize the efficiency of the simulation by reducing the number of time points that need to be evaluated. Finally, we provide the experimental results of our optimized algorithm measured on a cycle accurate simulator used in conjunction with a hardware emulator.

Categories and Subject Descriptors

B.6.3 [**Logic Design**]: Design Aids—*simulation*

General Terms

Algorithms, Verification

Keywords

Emulation, functional verification, oscillatory combinational loops

1. INTRODUCTION AND RELATED WORKS

Logic simulators are a key requirement in circuit design and verification flows nowadays. Logic design engineers are required to simulate their digital circuits to verify their expected functionality. Simulators can be used to verify the intended functionality at different levels, from behavioral to gate levels. Cycle accurate gate-level logic simulators are particularly critical in design and verification flows because the low-level specification and the fine granularity of the gates provide an accurate representation of the hardware functional behavior. Gate-level simulators are used to verify synthesized circuit netlists in different contexts. They can be used either as standalone verification tools or as embedded tools in FPGAs or hardware emulators [1] that combine faster functional verification with full visibility, which is the original motivation of this paper.

DAC'10, June 13-18, 2010, Anaheim, California, USA

Cycle accurate simulators use different delay models for digital circuit elements. Examples of delay models include zero-delay, ideal-delay and inertial-delay [2].

Simulators use different approaches to simulate combinational circuits [7]. In an *oblivious* simulator, all gates are simulated at each time unit. *Demand-driven* simulators are more efficient than *oblivious* simulators since the gates are evaluated only when their outputs are needed. In *event-driven* simulators only gates whose inputs have changed are evaluated. This requires complex dynamic schedulers compared with the simple static schedulers of *demand-driven* simulators.

The simulation process usually involves two main phases. In the first phase, the circuit netlist is compiled to be restructured and optimized. In the second phase, the compiled netlist is evaluated using the input stimuli specified in the testbench. The first phase is referred to as the *compiler* in this paper and the second phase as the *simulator*.

Combinational loops, which are logic gates (extended by including open-gate latches) with feedback signals can appear in a logic circuit for a variety of reasons. They can be found in asynchronous circuits, a sequential circuit might implement a RS-latch using cross-coupled NOR-gates, they can be created through latches, or generated by high-level synthesis tools to reduce the number of synthesized gates [8]. Combinational loops can be either potentially oscillatory or non-oscillatory (monotonic). Gupta, et al. [4] provided a method to convert monotonic loops into functionally equivalent acyclic logic. Shiple et al. [9] provided a procedure to determine statically whether a circuit produced a stable output sequence for every input sequence, under all possible circuit delays. If so, their procedure produced an equivalent acyclic circuit. Their work was built upon Malik's [6] analysis of cyclic combinational circuits. Hommais, et al. [5] described a cycle accurate simulator to handle non-oscillatory combinational loops which existed between Finite State Machines.

2. MOTIVATION

Simulation algorithms in the papers mentioned above report errors when an oscillatory loop is detected. However, post-synthesis cycle accurate simulators are required to evaluate oscillatory loops precisely in several scenarios. A designer might intentionally include combinational loops in a design to reduce the number of gates, only using the loop outputs when the loop is not oscillating [8]. Simulators that provide visibility to hardware emulators are also required to evaluate oscillatory loops accurately. When a digital circuit is emulated, the emulator records the inputs to the circuit at every clock cycle and the state values at intervals. Then, the visibility simulator calculates the state values at every clock cycle using the emulator recorded values. High capacity emulators are used to verify system-on-chip architectures. System-on-chip bus-based architectures can include arbitration schemes which are based on token-ring protocols implemented as

combinational loops. The combinational loop oscillates when there is not any request for the token in the ring. The cycle accurate simulator in this scenario is required to evaluate oscillatory loops precisely. The simulator is also required to report warnings when oscillation is detected to highlight oscillations that are due to the design errors.

As digital designs grow in size and complexity, it is a crucial objective to improve the performance of simulators. Oscillatory combinational loops can significantly degrade the performance of cycle accurate simulators, regardless of their type (*oblivious, event-driven, or demand-driven*). The following sections provide an algorithm that first detects oscillatory loops dynamically during simulation, once the inputs to the circuit have become stable. Then, the algorithm computes a multiple of the oscillation period. The oscillation period is defined as the number of time units between two identical states of the circuit (including outputs and delay elements). As long as the inputs to the circuit remain stable, the algorithm optimizes the simulation by skipping the evaluation of a multiple of oscillation periods.

The next section introduces our optimized algorithm for handling oscillatory combinational loops. Section 4 provides our experimental results. Our testing platform uses a cycle accurate *demand-driven* software simulator in conjunction with a hardware emulator. Section 5 provides conclusions.

3. THE ALGORITHM

An overview of the algorithm is as follows. The algorithm first determines whether the circuit is stable or oscillating, once its inputs reach stability. If the circuit oscillates then a multiple of the oscillation period is computed. Finally, the simulation is accelerated by skipping the evaluation of a number of time points corresponding to a multiple of the oscillation period previously computed.

The following subsections further explain how to determine dynamically whether a circuit is stable or oscillating. When a circuit contains oscillating loops, we will present a procedure and its proof to compute a multiple of its oscillation period. We will then show the pseudo-code of the algorithm that we propose to accelerate simulation of the circuit.

3.1 Determining stability or oscillation of circuit

In the general case, when the number of delay elements in a circuit is α, the maximum delay of any element is δ and the inputs to the circuit are stable, an upper bound for the circuit to reach either stability or oscillation is $\lambda = \delta(2^{\alpha} - 2)$ (Brzozowski et al. [2]). In asynchronous circuits, δ is the delay of the gate with the maximum delay and α is the number of gates with delays. In synchronous circuits, α is the number of state elements and δ is the delay of the state with the slowest clock.

In our cycle accurate simulator, once the circuit inputs become stable, L time-units are simulated. The stability upper bound L is calculated by the *compiler* as $L = K + D(2^n - 1)$ where n is the number of delay elements in the combinational loops of the circuit, D is the maximum delay of n elements, and K is the longest acyclic path delay of the circuit. Simulating K time-units after the circuit inputs become stable guarantees that the combinational loops inputs (excluding the feedback signals) become stable. Simulating $D(2^n - 1)$ time-units after the loops inputs become stable covers the space of all possible values for the delay elements in the loops. Therefore, L is an upper bound for the circuit to reach either stability or oscillation. In practice, the upper bound L is significantly smaller than λ because combinational loops are a small portion of circuits (i.e., $n \ll \alpha$).

A boolean variable $stability$ is defined to record whether the circuit is stable. The boolean is set to false when at least one delay element in the circuit is unstable. Otherwise, it is set to true. A delay element is unstable when its input and output values are different. The computation overhead associated with $stability$ is very small since it requires one comparison and one potential value assignment per delay element evaluation. If $stability$ is false after simulating L time-units, an oscillatory loop has been detected.

3.2 Determining a multiple of oscillation period

In general, the oscillation period of a combinational loop depends on its logic, input stimuli, delay values, and the delay model. A loop can be non-oscillatory or oscillate with different oscillation periods based on its inputs. Figure 1 shows an example of a circuit with two oscillation periods. $in1$ and $in2$ are the inputs of the circuit. $E1$ and

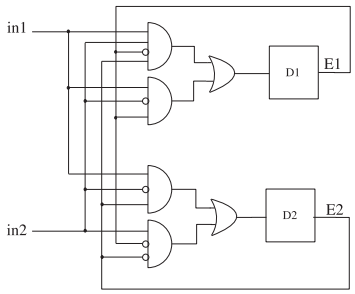

Figure 1: A circuit with two oscillation periods.

$E2$ are the outputs of two elements with delays of $D1$ and $D2$ time-units, respectively. The gates in the circuit have zero delay (i.e., the delay elements represent the delay of the gates). If $D1 = D2 = 1$ time-unit and the inputs have become stable after time t_0, the behavior of the circuit is as follows:

- Non-oscillatory ($in1 = in2 = 0$)
  $E1(t) = E2(t) = 0$ for $t \geq t_0$
- Memory ($in1 = 1, in2 = 0$)
  $E1(t+1) = E1(t)$
  $E2(t+1) = E2(t)$ for $t \geq t_0$
- Oscillation period 2 ($in1 = 0, in2 = 1$)
  $E1(t+1) = 0$
  $E2(t+1) = \sim E2(t)$ for $t \geq t_0$
  $E1E2 = \{00, 01, 00, 01, ...\}$ starting from t_0
- Oscillation period 3 ($in1 = 1, in2 = 1$)
  $E1(t+1) = \sim E1(t).E2(t)$
  $E2(t+1) = \sim E1(t). \sim E2(t)$ for $t \geq t_0$
  $E1E2 = \{00, 01, 10, 00, 01, 10, ...\}$ starting from t_0

For every input stimuli of a circuit, the oscillation period can be calculated either statically by the *compiler* or dynamically during simulation. Brzozowski et al. [2] and Burns [3] provided analytical solutions to find the oscillation period of combinational loops based on a directed graph representation of the circuit. The directed graph is processed to determine what sequence of inputs can produce oscillation. For every oscillation scenario, the oscillation period is calculated. Analytical approaches are practical and efficient for small circuits only. When the size of a circuit reaches millions of gates, the analytical solution cannot be applied due to the excessive amount of memory and processing time required (i.e., the complexity of analytical approaches increases exponentially with the number of delay elements and inputs to the combinational loops while the complexity of the proposed algorithm increases linearly with the number of delay elements in the combinational loops).

The oscillation period can efficiently be determined dynamically during simulation. In the ideal delay model case, a method for finding the oscillation period is as follows: Every element with delay D

is replaced with D concatenated elements with delay 1. Once the circuit inputs become stable, L (upper bound for stability) time-units are simulated. Then, a snapshot of the single-delay element values of the combinational loop is taken. The values of the single-delay elements are compared with the snapshot values at every time-unit. When the values are the same, the oscillation period is determined.

The inertial delay model is not in the scope of this paper. The remaining sections of this paper propose an efficient algorithm to determine the oscillation period when the delay elements are pipeline flops (which are a usual representation of delay elements). This algorithm does not require the circuit modifications described in the ideal delay model method above.

A well-known strategy for a *compiler* is to insert pipeline flops between combinational gates, if necessary, after timing analysis. Pipeline flops are also inserted to break combinational loops. Clock signals of pipeline flops can be several orders of magnitude faster than the fastest clock of the original design. Pipeline flops may have different delays (represented by a non-negative integer number of time units). The clock cycle duration of every state element (including pipeline flops) is assumed to be greater than the delay of combinational gates feeding the state input based on the guidelines provided by the timing analysis. As a result, simulators can accurately simulate a post synthesis netlist using zero-delay semantics.

A multiple of the oscillation period (p) can be computed, as indicated in the following pseudo-code, by comparing unstable state values at every M time-units where M is the least common multiple (LCM) of all unstable state delays.

```
Snapshot S = E(t)
p = 0;
do
   simulate M time-units;
   p = p + M;
while (S != E(t+p))
```

Let's assume:

I. Circuit $C = (E(t), E(t_0), I(t), O(t), T_E, T_O)$ has n elements with delay,

II. $E(t) = \{E_1(t), E_2(t), ..., E_n(t)\}$ where $E_i(t)$ is the value of element i at time t,

III. $I(t)$ and $O(t)$ are the set of input and output values of the circuit C, respectively,

IV. $T_E : I \times E \rightarrow E$, and $T_O : I \times E \rightarrow O$ are combinational logic transformation functions,

V. All inputs to the circuit C are stable after time t_0 (i.e., $I(t) = I(t_0)$ for $t > t_0$),

VI. C is either in a stable or oscillating state (non-transient state),

VII. M is the least common multiple (LCM) of all delays, and

VIII. A zero-delay semantics model, thus the delay elements in $E(t)$ are a set of unstable pipeline flops.

Theorem: p is a multiple of the oscillation period if $E(t) = E(t+p)$ where $p > 0$, $t \geq t_0$, and $(p\%M = 0)$.

Proof: Let's first prove the theorem for a circuit where $M = 1$ (i.e., all delays are equal to one). The lemma below proves that the values of unstable states during one oscillation period are unique when $M = 1$. Therefore, the number of time units between two identical set of state values is a multiple of the oscillation period.

Lemma: $E(t+1) = E(t+p+1)$ if $E(t) = E(t+p)$, $p > 0$, $t \geq t_0$, and $M = 1$.

We have:

$E(t+1) = T_E(I(t+1), E(t))$ for $M = 1$, by definition of T_E, hence

$E(t+1) = T_E(I(t+1), E(t+p))$ by hypothesis

$E(t+1) = T_E(I(t+p+1), E(t+p))$ since $t \geq t_0$, $t+p > t_0$, and inputs are stable

$E(t+1) = E(t+p+1)$.

This completes the proof of the lemma. Similarly, we can also prove that $O(t+1) = O(t+p+1)$.

To complete the proof of the theorem for $M = 1$, let's now consider the sequence of values of delay elements and outputs, starting from time t:

$\{EO(t), EO(t+1), ..., EO(t+p-1)\}$,

$\{EO(t+p), EO(t+p+1), ..., EO(t+p+p-1)\}$

where $EO(t) = (E(t), O(t))$.

From the Lemma, we have:

$EO(t+i) = EO(t+p+i)$ for $i = 0$ to $p-1$ when $E(t) = E(t+p)$.

Therefore, p is a multiple of the oscillation period.

Note that the Lemma is not always correct if $M \neq 1$. For instance, consider a simple combinational loop with only one inverter element E_1 with a delay of two. The circuit oscillates with an oscillation period of four as follows: $E1 = \{0, 0, 1, 1, 0, 0, 1, 1, ...\}$ starting from t_0. We have $E_1(t_0) = E_1(t_0 + 5) = 0$ but $E_1(t_0 + 1) = 0 \neq E_1(t_0 + 5 + 1) = 1$.

Let's now finalize the proof of the theorem for $M \neq 1$. A circuit with arbitrary delay elements can always be transformed into a functionally equivalent circuit with delays of one (i.e. $M = 1$). Figure 2 shows an example. Figure 2a shows element E_1 (a pipeline flop) with

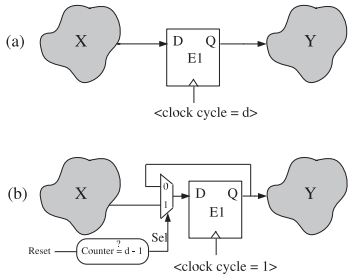

Figure 2: Transformation of an element with delay d to 1. (a) state element with delay d. (b) state element with delay 1.

a delay of d. Its input is the output of logic X and its output is an input to logic Y. The delay of E_1 can be reduced to one without changing the functionality of the circuit by adding a zero-delay multiplexer to its input (Figure 2b). Input 0 of the multiplexer is the output of E_1 and input 1 is the output of X. The select of the multiplexer is generated by synchronous logic. This logic has a reset input and a modulo-d counter. The single output of this logic (select of the multiplexer) is "1" only if the output of the counter is equal to $d-1$:

$$Sel = \begin{cases} 1 & if\ (counter = d-1) \\ 0 & otherwise \end{cases}$$

The modulo-d counter can be implemented using $\lceil log(d) \rceil$ number of state elements with a delay of one. Other combinational elements have zero delays.

Let $E'(t)$ be the set of values of all state elements in the counter after the circuit transformation. Since the delay of all unstable states

is one in the transformed circuit ($M' = 1$), we have already proved that: p is a multiple of the oscillation period if $E(t) = E(t+p)$ and $E'(t) = E'(t+p)$ where $p > 0$, and $t \geq t_0$. The second condition of the if statement $(E'(t) = E'(t+p))$ is always true when p is a multiple of the LCM of all delays of the original circuit (M). Consequently, p is a multiple of the oscillation period if $E(t) = E(t+p)$ where $p > 0$, $t \geq t_0$, and $(p\%M = 0)$. This completes proof of the theorem.

3.3 Pseudo code

Based on the results established above, we will now provide the pseudo-code of the algorithm proposed to efficiently simulate circuits containing oscillating loops.

If the circuit has been determined to oscillate after L time-units, the algorithm takes a snapshot of all unstable state values (pipeline flops). The values of the unstable states are compared to the snapshot values at every M time-units, where M is the LCM of all unstable state delays. As proven above, p is a multiple of the oscillation period and thus, as long as the circuit inputs remain stable, the simulation can be accelerated by skipping the evaluation of a multiple of p time-units.

```
Simulate L time-units
If circuit is oscillating
  S = current value of unstable states
  p = 0;
  do
    Simulate M time-units
    (where M is LCM of delays)
    p = p + M;
  While (unstable states values != S)
  Skip simulation of a multiple of p
```

As an example, let's now consider the circuit in Figure 1. Let's assume the delay elements $E1$ and $E2$ are synchronized pipeline flops with delays $D1 = D2 = 2$ time-units. The inputs have become stable after time t_0 with values $in1 = in2 = 1$. The circuit oscillates with an oscillation period of six as follows:
$E1E2 = \{00, 00, 01, 01, 10, 10, 00, 00, ...\}$ starting from t_0
The snapshot $S = (00)$ is taken at t_0. The LCM of the delays is two. Therefore, the algorithm compares the snapshot with $E1E2$ at every two time-units. After three comparison, the algorithm returns six as the oscillation period.

4. EXPERIMENTAL RESULTS

The algorithm was implemented in C++ language. It was integrated into a cycle accurate simulator which was used to provide visibility for hardware emulation. A circuit, described by VHDL, Verilog or System Verilog is mapped onto a FPGA by the *compiler* (synthesis tool). The FPGA models logic gates with 4-input Lookup Tables (LUTs). The *compiler* calculates the upper bound L for the delay which is required for the circuit to reach either stability or oscillation based on the longest path delay of the circuit and static analysis of the combinational loops.

Table 1 presents the simulation results of evaluating five circuits with oscillatory combinational loops. The number of simulation cycles for every circuit is one million cycles of the fastest clock. "LUTs" and "FFs" are the number of lookup tables and flip-flops in the circuits, respectively. The row "TU/cycle" provides the number of time-units per fastest clock cycle of the circuits. The row "L" shows the upper bound in time-units calculated by the *compiler* for the circuits to reach either stability or oscillation. The row "period" shows a multiple of the oscillation period computed by the algorithm during simulation. The simulation runtime is presented for two scenarios, without ("no-opt") and with ("opt") our optimization algorithm. The last row ("speed up") presents the overall improvement in runtime. The improvements shown apply to the total simulation time for the entire design.

Table 1: Simulation time improvement for circuits with oscillatory combinational loops

circuit	1	2	3	4	5
LUTs	64	130	593	66104	66482
FFs	75	134	824	13479	21981
TU/cycle	2k	1.5k	3k	5k	10k
L	12	26	7	15	24
period	2,3	12	3	2	2
no-opt	271s	516s	525s	4440s	41000s
opt	152s	369s	386s	960s	1200s
speed up	1.8	1.4	1.4	4.6	34

The results are sensitive to the circuit clock frequency, the number of oscillating loops in the design, and the number of cycles when oscillatory behavior occurs. The acceleration observed will increase when the clock frequency decreases (since the number of oscillations per cycle will increase). Note that certain situations can require setting clock frequencies low, for instance in case of in-circuit emulator configurations connected to external targets which might have specific timing constraints. Therefore, the overall improvement in runtime can vary from a small number to several orders of magnitude in extreme scenarios.

5. CONCLUSIONS

We introduced an efficient algorithm for post-synthesis logic simulation of digital circuits with oscillatory combinational loops. The algorithm is able to detect oscillatory loops dynamically, once the inputs to the circuit remain stable. We provided a novel approach to find a multiple of the oscillation period during simulation which was then used to accelerate the simulation by skipping the evaluation of a multiple of the oscillation period, as long as the inputs to the circuit remain stable.

6. REFERENCES

[1] *Mentor Graphics Veloce Hardware Emulator*. http://www.mentor.com/products/fv/emulation-systems/veloce.

[2] J. A. Brzozowski and C.-J. H. Seger. *Asynchronous Circuits*. Springer-Verlag, 1995.

[3] S. M. Burns. *Performance Analysis and Optimization of Asynchronous Circuits*. PhD thesis, California Institute of Technology, 1991.

[4] A. Gupta and C. Selvidge. Acyclic modeling of combinational loops. In *International conference on Computer-Aided Design*, pages 343–347, 2005.

[5] D. Hommais and F. Petrot. Efficient combinational loops handling for cycle precise simulation of system on a chip. In *Euromicro Conference*, volume 1, pages 51–54, August 1998.

[6] S. Malik. Analysis of cyclic combinational circuits. *IEEE Transaction on Computer-Aided Design*, pages 950–956, July 1994.

[7] J. A. Payne. *Introduction to Simulation: Programming Techniques and Methods of Analysis*. McGraw-Hill, 1982.

[8] M. D. Riedel and J. Bruck. The synthesis of cyclic combinational circuits. In *Design Automation Conference*, pages 163–168, Anaheim, CA, 2003.

[9] T. R. Shiple, G. Berry, and H. Touati. Constructive analysis of cyclic circuits. In *Proceedings of European Design and Test Conference*, pages 328–333, Paris France, March 1996.