Identification of Critical Paths in Circuits with Level-Sensitive Latches

Timothy M. Burks Karem A. Sakallah Trevor N. Mudge

The University of Michigan

Abstract

This paper describes an approach to timing verification of circuits with level-sensitive latches which focuses on the critical paths that constrain the operating speed of these circuits. The timing model we use has been referred to as the SMO model and was originally described in [1]. In this work, we show that three types of critical paths can arise in the SMO formulation; verifying their timing is sufficient to ensure correct operation. We present an algorithm for identifying these paths and discuss its relationship to other approaches to solving the SMO model equations. Finally, we present results which demonstrate our algorithm on circuits from the ISCAS89 benchmark suite.

1 Motivation

A number of methods for timing verification have been recently published that are based on solving the constraints of the SMO model [1, 2, 3]. The *check*T_c program [1] rapidly finds solutions by iteratively solving the arrival and departure time equations. However, as observed by Szymanski [2], *check*T_c suffered from unbounded run times in the presence of critical loops that were marginally longer than the clock cycle allowed. With a simple modification, Szymanski showed that these loops could be detected in a number of iterations bounded by the number of latches in the circuit.

In this paper, we present precise definitions of the critical paths that arise in circuits with level-sensitive latches. These critical paths are more complex than those arising in circuits controlled by flip-flops because they may flow through one or more transparent latches. We also present an algorithm which identifies timing violations and produces a list of all associated critical paths. The algorithm is based on the iterative expansion of the most critical paths in a circuit.

2 Critical Path Definitions

2.1 Timing models

Our analysis is based on the model presented in [1]. For reference, the model equations and constraints are listed in Table 1. Variables in the clock model describe cycle time (T_c), phase width (T_p), and the ending times of each clock phase (e_p). Parameters in the circuit model include latch setup and hold times (S_i and H_i), clock phase of each latch (p_i), minimum and maximum delays through each latch (δ_i and Δ_i), and minimum and maximum combinational delays between each connected pair of latches (δ_{ij} and Δ_{ij}).

Clock Constraints	
$T_p \geq w$	(1)
$T_c - T_p \geq w$	(2)
Combinational Propagation Constraints	
$a_i = \min_{j=1,n}(d_j + \delta_j + \delta_{ji} - E_{p_j p_i})$	(3)
$A_i = \max_{j=1,n}(D_j + \Delta_j + \Delta_{ji} - E_{p_j p_i})$	(4)
Latch Macromodels	
$A_i \leq T_c - S_i$	(5)
$a_i \geq H_i$	(6)
$d_i = \max(a_i, T_c - T_{p_i})$	(7)
$D_i = \max(A_i, T_c - T_{p_i})$	(8)
Phase Shift Operator	
$E_{p_j p_i} = T_c - ((e_j - e_i) \bmod T_c)$	(9)

Table 1: Timing Model Summary

We model signals at the input of each latch with the range of possible times (a_i, A_i) within which a signal can arrive at the input. Signals at the output of a latch are described with the earliest and latest times (d_i, D_i) that

0-8186-3010-8/92 $03.00

new signal values can depart from the output. All arrival and departure times are defined in a frame-of-reference that begins and ends on the clock edge that closes their corresponding latch. The phase-shift operator $E_{p_j p_i}$ is used to convert signal times from the frame-of-reference of latch j to that of latch i. The requirement that $0 < E_{p_j p_i} \leq T_c$ guarantees that signals departing from latches controlled by phase p_j will arrive at latches controlled by phase p_i before the next closing edge of ϕ_{p_i}.

2.2 Critical Paths

In this section we define three types of critical paths which correspond to the constraints implied by the SMO model. A critical path is a section of a circuit which most severely constrains the speed at which the circuit can be run. In circuits clocked with edge-triggered flip-flops, critical paths are very simple; a path between flip-flops i and j is critical if the time between the closing edge of ϕ_{p_i} and the closing edge of ϕ_{p_j} is less than the time required for a signal to propagate along the path and be properly set up at the input of flip-flop j. For level-sensitive latches, the ability of signals to flow directly through a latch can allow critical paths which are significantly more complex. Critical paths still begin and end at latches, but other latches can lie within the path if they allow signals to flow directly through them. In addition to the setup-constrained critical paths, two other types of critical paths arise, due to the hold constraints on latches and a cyclic constraint on loops of flow-through latches.

A path P is formally defined as a sequence of latches

$$P = L_0 \rightarrow L_1 \rightarrow \ldots \rightarrow L_m$$

where each latch is directly connected to its predecessor through a combinational logic segment (Fig. 1-a). The length of P is defined as the number of combinational segments in the path, $|P| = m$.

We require that all paths begin on a clock edge because of the following assumption: in a closed synchronous system (free of unsynchronized inputs or outputs), no asynchronous feedback loops will be present, and the only stimuli that the system will receive will be from events in the clock schedule.

2.2.1 Critical Long Paths: A critical long path is defined as a path for which an increase in any delay along the path causes, or worsens, a setup time violation at the input of the last latch in the path. Formally, a path is a critical long path if and only if it satisfies the following constraints (Fig. 1-c):

$$D_0 = T_c - T_{p_0} \tag{10}$$

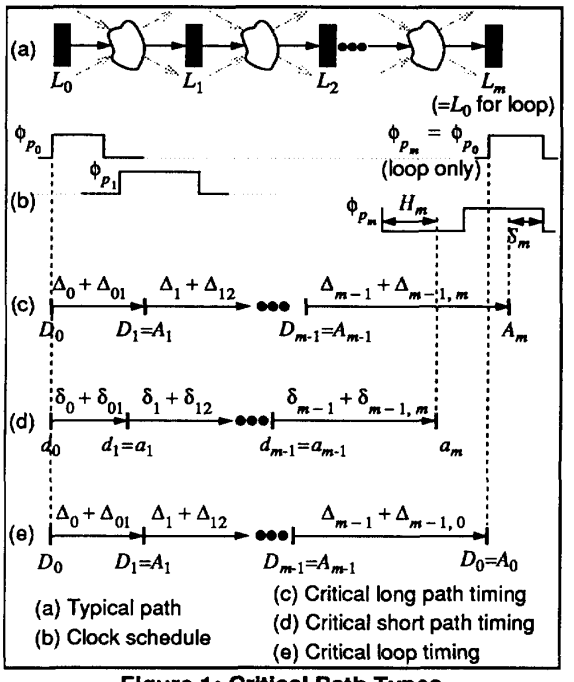

Figure 1: Critical Path Types

$$A_i = D_{i-1} + \Delta_{i-1} + \Delta_{i-1,i} - E_{p_{i-1}p_i},\ \forall i \in \{1,\ldots,m\} \tag{11}$$

$$D_i = A_i,\ \forall i \in \{1,\ldots,m-1\} \tag{12}$$

$$A_m \geq T_c - S_m \tag{13}$$

A signal on a critical long path begins to propagate on the opening edge of the clock phase controlling L_0 (10). It then provides the critical late arrival time at each combinational and sequential device along the path (11) and must flow through any intermediate latches on the path (12). Finally, a critical path must be tight on (or violate) the setup constraint at the end of the path (13).

2.2.2 Critical Short Paths: A critical short path is defined as a path for which a decrease in any delay along the path causes, or worsens, a hold time violation at the input of the last latch in the path. Formally, a path is a critical short path if and only if it satisfies the following constraints (Fig. 1-d):

$$d_0 = T_c - T_{p_0} \tag{14}$$

$$a_i = d_{i-1} + \delta_{i-1} + \delta_{i-1,i} - E_{p_{i-1}p_i},\ \forall i \in \{1,\ldots,m\} \tag{15}$$

$$d_i = a_i,\ a_i > T_c - T_{p_i},\ \forall i \in \{1,\ldots,m-1\} \tag{16}$$

$$a_m \leq H_m \tag{17}$$

Like the critical long paths, a critical short path begins on a clock transition (14). Signals flowing along the path must provide the critical early arrival time at each device (15) and signals must flow through any intermediate latches in the path (16). The arrival time at the end of the path must be tight on (or violate) the final hold time constraint (17).

2.2.3 Critical Loops: The critical long and short paths described in the previous sections directly correspond to the setup and hold constraints in Table 1. However, there is also an additional constraint that arises whenever there are cycles in a circuit. It can be derived from relations (4) and (8) and requires that for each latch in the cycle, $A_i \le D_i$. This reflects the assumption that signals propagating around a cycle will arrive back at their starting point in time to begin a new cycle no later than the previous time they began propagating. A similar constraint exists for the early time variables ($a_i \le d_i$) but can be shown to be subsumed by the late signal constraints.

The critical paths that correspond to these constraints are called *critical loops*, and they are defined as a circular path in a circuit containing m latches, numbered 0 to m, with $L_m = L_0$, and for which an increase in any delay around the loop would either cause a signal to arrive *after* its required departure time or cause an already-present loop violation to worsen.

Formally, $P = L_0 \rightarrow \ldots \rightarrow L_{m-1} \rightarrow L_m$ is a critical loop if and only if $L_m = L_0$ and the following constraints are satisfied (Fig. 1-e):

$$D_0 = T_c - T_{p_0} \tag{18}$$

$$A_i = D_{i-1} + \Delta_{i-1} + \Delta_{i-1,i} - E_{p_{i-1}p_i},\ \forall i \in \{1,\ldots,m\} \tag{19}$$

$$D_i = A_i,\ \forall i \in \{1,\ldots,m-1\} \tag{20}$$

$$A_0 \ge D_0 \tag{21}$$

3 Critical Path Verification

A path which satisfies the first three constraints in any of the path definitions in Sec. 2.2 is called a *candidate* path. Depending on its setup (13), hold (17), or loop (21) constraint, a candidate path can be either *satisfied* (with no slack on its final constraint), *noncritical* (with positive slack), or *violated* (with negative slack).

It is straightforward to show that if all the candidate paths in a circuit are satisfied or noncritical, the timing constraints of Table 1 are guaranteed to be satisfied. Figure 2 sketches an algorithm that identifies all the candidate long paths and loops in a circuit. The algorithm which identifies candidate short paths is similar and is omitted due to space considerations.

```
create an initial null candidate path at the output of each latch
propagate paths to the inputs of all fanout latches, marking them active
while there are active paths
    for each latch with an active path on its inputs
        identify the latest arriving signal(s) on the latch inputs
        if (this signal is active and
         the signal is not blocked by the clock and
         the associated path is not a cycle) then
            extend the path to include the current latch
            propagate it on the latch outputs and mark it active
        end if
        deactivate all signals and paths on the latch inputs
    end for
    propagate signals from latch outputs to inputs of fanout latches
end while
for each latch in the circuit
    check timing constraints on input signals
end for
```

Figure 2: Long Path/Loop Verification Algorithm

Each iteration in the algorithm generates successively longer candidate paths, with the i-th iteration generating all the candidate paths of length i. This is done by extending paths from iteration i-1. The extensions are performed by identifying the most critical input to each latch and if the associated input path is active, the path is extended as long as (1) the resulting extended path is not a cycle, and (2) signal flow through the latch is not blocked by the clock. If parallel candidate paths are present, all are extended into successive iterations. Paths are deactivated at the end of each iteration to ensure that only paths of length i-1 are extended in iteration i.

The following theorem guarantees that all candidate paths can be generated in an iterative manner:

Theorem 1 If a path $P = L_0 \rightarrow \ldots \rightarrow L_{m-1} \rightarrow L_m$ is a candidate path, then $P' = L_0 \rightarrow \ldots \rightarrow L_{m-1}$ is also a candidate path.

As a result, we know that all of the candidate paths of length m can be generated as extensions of candidate paths of length m-1.

When there are no more active paths, the set of candidate paths is checked for possible setup, hold, and cyclical timing errors.

3.1 Performance Issues

The number of iterations required by the algorithm is bounded by the following two lemmas:

Lemma 2 It is sufficient to verify only candidate paths that contain no internal cycles.

Lemma 3 The longest cycle-free candidate path in a circuit can be no longer than $|L|$, where $|L|$ is the number of latches in the circuit.

Thus the algorithm will require no more than $|L|$ iterations. Each iteration involves examining as many as the $|E|$ edges which interconnect the latches in the circuit. This would

suggest that the complexity of the algorithm is O(|L||E|). However, there is an additional hidden cost to this algorithm: since each iteration of the algorithm involves a test to determine whether a candidate path is a cycle, the computational cost of each iteration may increase exponentially in the presence of many parallel candidate paths at latch inputs. However, we believe that for most practical circuits critical paths are quite short and this exponential cost can be bounded by a relatively small constant (call it k).

The strength of this algorithm lies in its expected-case performance and not its worst-case complexity. After $|P_{max}|$ iterations the algorithm will have generated all of the candidate paths for a circuit, where P_{max} is the longest candidate path in the circuit. This practical bound on path length makes the algorithm $O(|P_{max}||E|k)$ and for most circuits $|E|$ is much less than its worst-case value ($|L|^2$) making the actual expected performance somewhat better than quadratic in the number of latches.

3.2 Detecting start-up errors

Because timing checks are not performed until all signal arrival times have converged, the verification algorithm as shown here only checks for errors in the steady-state timing of a circuit. However, the algorithm can also be viewed as a simulation of the circuit's start-up behavior. The i-th iteration thus calculates the signal arrival and departure times occurring in the i-th clock cycle since the circuit was started. Only one type of error can arise during start-up and not during normal operation: transient hold violations. This will only happen when the hold constraint for a latch can only be satisfied by a critical short path whose length is greater than 1. The start-up operation creates an arrival which is earlier than the steady-state arrival time, causing a temporary error. In the current implementation, we assume that these transient errors should be ignored, much as we ignore the data flowing through a pipeline before it has reached steady state. If desired, they could easily be detected by checking for hold violations during each iteration of the path extension algorithm.

3.3 Comparison with *checkT$_c$*

A simpler algorithm for detecting timing errors is used in the *checkT$_c$* program [1]. A detailed analysis of the algorithm is presented in [2]. *checkT$_c$* begins by assuming that all signals depart at their earliest possible times, i.e. the rising edge of the clock for positive level-sensitive latches. It then iteratively recalculates these departure times until they converge to a final solution, which can then be checked for setup and hold violations. Loop constraint violations cause the departure times around the loop to increase without bound, and can be eventually detected as setup violations.

The *checkT$_c$* algorithm has been observed to converge to solutions rapidly. If no critical loops are present, *checkT$_c$* can quickly calculate arrival times and critical long and short paths can be extracted using the definitions described in Sec. 2.2. Also, the *checkT$_c$* algorithm can similarly benefit from the typical limits on path lengths occurring in practical circuits.

The algorithm that we present here thus has one primary advantage over the *checkT$_c$* algorithm: its expected performance in the presence of critical loops. *checkT$_c$* detects violated critical loops by waiting for the departure times around the loop to increase until a setup time violation appears. This caused the original worst-case performance of the *checkT$_c$* algorithm to be poor, particularly when the sum of the delays around a critical loop was slightly greater than the total time available for signals to propagate around the loop. However, as described in [2], the late signal calculations in the *checkT$_c$* algorithm can (with slight modifications) be viewed as a special case of the Bellman-Ford algorithm; in this case critical loops can also be detected if arrival times have not converged after $|L|$ iterations of the algorithm. Since each iteration of the algorithm must examine up to all $|E|$ edges in a circuit, the modified version of *checkT$_c$* can be guaranteed to run in at worst $O(|L||E|)$ time.

4 Experiments

The algorithm described in Fig. 2 was implemented in C++ as an application layer around an object-oriented framework for CAD tool prototyping under development at the University of Michigan. To test our algorithm and implementation, we analyzed circuits taken from the ISCAS89 sequential benchmarks [4]. Using a unit delay model, we performed both single- and two-phase level sensitive timing verifications. In all of our verifications we assumed the use of symmetric clock phases with 50% duty cycles. For the two-phase circuits, the clock phases were complements of one another.

Latch-to-latch delays were obtained in a preprocessing step that found the shortest and longest paths between each pair of latch output and input pins. A single-phase timing verification was then performed for the benchmark circuits clocked with latches at a cycle time equal to the minimum flip-flop cycle time. Circuit inputs were assumed to arrive and outputs were assumed to be latched on the falling clock edge. As could be expected, the verification algorithm found a large number of critical short paths that made it impossible to operate the circuit at the desired cycle time. In fact, virtually every latch input and

circuit output pin had a hold time violation. The list of associated critical short paths could be used, however, to guide a subsequent padding of delays.

The two-phase circuits were generated according to the method described in [5] in which all input and output pins were replaced with latches, the circuit was duplicated and latch inputs and outputs were cross-connected between the original circuit and its duplicate. For each circuit, the minimum cycle time ($T_{c,min}$) was determined using $minT_c$ [1] and the verifier was run at 100% and 90% of the minimum cycle time. Table 2 contains CPU times observed for each verification along with the maximum candidate path lengths and the number of setup and loop violations found for each of the $0.9T_{c,min}$ verifications. Since critical loops can be reported once for each latch in the loop, the number of unique critical loops is a fraction of the value shown.

Performance numbers include time required to parse circuits and a significant amount of overhead introduced by the prototyping environment. However, their dependency on problem size can be observed to lie between linear and quadratic. Also, running times can be seen to increase as the longest observed paths increase, reflecting the cost of searching back along these paths to detect cycles.

Finally note that the candidate path lengths which we observed were relatively short, less than or equal to 5 for the error-free circuits. This explains the rapid convergence of $checkT_c$ iterations and supports the average-case complexity analysis in Sec. 3. Furthermore, it is interesting to note that the number of examined paths increase as the cycle time is reduced. The number of candidate paths and their maximum length both increase monotonically as cycle time is reduced. This is expected since as the cycle time decreases, more signals arrive late enough to flow through transparent latches.

5 Conclusions and Future Plans

This work has defined the three types of critical paths which can exist in a latch-controlled circuits, and work was presented that shows that verifying the timing of these paths is sufficient to ensure that a circuit will work subject to the constraints presented in [1]. In studying our initial implementation, we observed that a number of redundant path expansions remained in our algorithm, so we are developing additional heuristics which could be used to further prune the paths which are extended. We are also developing methods for finding the minimum cycle time of a circuit from path information (noting similarities to the approach in [5]), and we are working to streamline our prototype implementation.

Acknowledgments

This work was supported in part by NSF Grant MIP-9014058. T. Burks was supported by a DoD NDSEG fellowship.

References

[1] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "$checkT_c$ and $minT_c$: Timing verification and optimal clocking of synchronous digital circuits," *ICCAD-90 Digest of Technical Papers*, November 1990.

[2] T. G. Szymanski, "Verifying Clock Schedules," *ICCAD-90 Digest of Technical Papers*, 1992.

[3] N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "A pseudo-polynomial algorithm for verification of clocking schemes," *ACM/SIGDA Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, March 1992.

[4] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," *ISCAS89 Proceedings*, 1989.

[5] T. G. Szymanski, "Computing Optimal Clock Schedules," *Design Automation Conference Proceedings*, 1992

circuit	1phase latch count	1phase flip-flop $T_{c,min}$	# of short path violations	2phase latch count	2phase $T_{c,min}$	$T_{c,min}$ longest cand. path	$T_{c,min}$ cpu time	$0.9\,T_{c,min}$ longest cand. path	$0.9\,T_{c,min}$ cpu time	$0.9\,T_{c,min}$ # of setup violations	$0.9\,T_{c,min}$ # of loop violations
s27	3	6	4	16	8	3	0.2	3	0.2	0	2
s344	15	20	26	70	28	3	1.0	4	1.1	0	10
s400	21	9	26	60	12	5	1.5	5	1.5	6	2
s953	29	16	49	136	26	2	2.5	5	3.6	0	14
s838	32	17	33	134	17	1	5.7	1	6.3	2	0
s9234	228	58	246	538	76	2	31.9	3	40.3	0	16
s13207	669	59	781	1642	92	3	59.3	3	60.2	0	18
s38584	1452	56	1719	3484	70	3	459.5	7	573.9	2	88

Table 2: Representative ISCAS Benchmark Results