# An Advanced Timing Characterization Method Using Mode Dependency

Hakan Yalcin[1], Robert Palermo[1], Mohammad Mortazavi[1], Cyrus Bamji[2], Karem Sakallah[3], John Hayes[3]

[1]Cadence Design Systems
San Jose, CA 95134
{hyalcin,bobp,mortazav}@cadence.com

[2]Canesta Inc.
Santa Clara, CA 95054
bamji@home.com

[3]Dept. of Electrical Eng. and Computer Science, University of Michigan
Ann Arbor, MI 48109
{karem, jhayes}@eecs.umich.edu

## ABSTRACT

*To address the problem of accurate timing characterization, this paper proposes a method that fully exploits mode dependency. It is based on the premise that circuit delays are determined largely by a set of control inputs for which the number of useful combinations, i.e., modes, is small for most practical circuits. We take the mode-dependent characterization approach further and enhance it so that the delays of the I/O paths between the control inputs and outputs are calculated more accurately. We prove that, with a careful choice of propagation conditions, our method can generate timing models with very tight path delays that are guaranteed to give correct results. Experimental results using real-life circuits show that circuit delays can vary significantly among different modes for both control and data input delays, and capturing this variation can have a significant impact on the overall system timing.*

## 1. INTRODUCTION

An important problem in a hierarchical design and verification strategy is accurate timing characterization of circuit blocks making up a large system. A timing model for each block is to be calculated**,** which can then be used in place of the block in subsequent timing analyses of the system. This may be required for efficiency reasons, for instance, when the block is instantiated more than once, or for IP protection reasons, as the IP provider may not want to reveal the contents of the block. In most cases, no assumption can be made about the environment in which the block is instantiated. For instance, the input arrival times cannot be predicted, so the timing model must be correct for any set of input arrival times. Ideally**,** using the model should yield the same results as if the block itself were present. But this requirement is usually hard to meet, so reasonable approximations are accepted in practice. The only strict requirement is that the timing model not yield more optimistic results.

Topological analysis is commonly used for characterization purposes, and easily satisfies the above requirement. But it has the serious deficiency of ignoring false paths, which are signal paths that are never activated during actual operation. Thus topological analysis can lead to overly pessimistic timing models. Unfortunately, detecting false paths is difficult, and requires making use of circuit functionality. A full-fledged functional analysis, using any of the existing path propagation conditions [1-3,5,7], can be computationally expensive. For instance, the approach by Kukimoto and Brayton [4] can in fact detect all false paths as it calculates I/O delays accurately for each input vector. However, a direct application of this method is difficult in practice since the number of input vectors for actual circuits can be extremely high.

In order to handle large circuits practically, a solution was proposed in [8] which takes advantage of *mode dependency.* This work views a circuit as having a number of modes of operation and assumes the circuit mode is determined by the circuit's control inputs. It then determines the I/O delays for each mode separately. This method is an improvement over topological analysis and can detect false paths due to control input dependencies. But it applies mode analysis only to the data inputs. For the control inputs, it continues to use topological analysis, which limits its accuracy. Also, in both [4] and [8], modeling of sequential circuits was not considered.

As we show in this paper, path delays for the control inputs can also vary significantly. For more accurate timing analysis, it is important to also capture the delay variation for control inputs. The method proposed in this paper precisely addresses this problem. Through a detailed analysis of event propagation conditions, we show that mode dependency can be exploited to determine accurate delays for both the data and the control inputs. Thus our method, which is called AdvChar, generates timing models with very tight delays while still satisfying the correctness requirement for the timing model. Furthermore, we extend AdvChar to handle sequential circuits based on a modeling technique by Venkatesh et al. [6].

The paper is organized as follows. Section 2 reviews the mode-dependent characterization approach and describes AdvChar for combinational circuits. Characterization of sequential circuits is covered in Section 3, followed by experimental results in Section 4.

## 2. MODE-DEPENDENT CHARACTERIZATION

The key assumption made in mode-dependent characterization is that the path delays of a circuit are determined**,** to a large extent**,** by its control inputs. The circuit to be characterized is assumed to operate in a number of modes, where each mode corresponds to a particular combination of the control inputs. The control inputs and the mode information are specified by the user. It is assumed the number of modes for which the circuit is to be characterized is small, which is often true in practice.

We next review the characterization method called ModeChar that was proposed in [8] and later elaborated in [9]. We then introduce our AdvChar method that enables the calculation of more accurate timing models.

### 2.1 The ModeChar Method

ModeChar constructs the timing model of a circuit in two steps: the first step calculates the I/O delays between the data inputs and the outputs, while the second step calculates the I/O delays between the

control inputs and the outputs. In the first step, the worst-case delay for each data input/output pair is calculated for each circuit mode as follows. First, the control input values (constants) corresponding to the current mode are applied to the circuit and are propagated as far as possible. Certain paths are blocked by controlling values that have propagated to their side inputs. Then, the worst-case delay between each data input and each output of the circuit is calculated by performing a topological analysis of the unblocked portion of the circuit. Therefore ModeChar effectively uses static sensitization for the data inputs.

In the second step, ModeChar simply uses topological analysis to calculate the worst-case delay between each control input and each output. The control input delays therefore apply to all the modes. The way ModeChar works is illustrated by the example circuit of Fig. 1. Suppose that *C* is a control input, *A* and *B* are data inputs and that all gates have unit delay. ModeChar yields the following timing model:

| Path | *C* = 0 | *C* = 1 |
|---|---|---|
| *A* => *Z:* | 6 | 3 |
| *B* => *Z:* | 2 | none |
| *C* => Z: | 5 | 5 |

By using the circuit's mode information, ModeChar is able to eliminate some false paths starting at a data input. For instance, path *A-p-q-r-s-t-u-Z* in the example circuit, which has a length of 7, is found to be false. Although this path can be activated under certain input arrival time conditions, it was proven in [8] that the use of topological delays for the control inputs ensures that the arrival times at the outputs are correctly calculated under any input arrival time condition. However, ModeChar has the following shortcomings: (1) it cannot detect any false path from the control inputs, (2) it completely ignores the mode dependency of the control inputs, which can be significant in practice.

## 2.2 The AdvChar Method

The conservative topological analysis method for handling control inputs in ModeChar brings up the question of whether delay calculation can be improved for these inputs. We answer this question positively by showing that the control input paths can, in fact, be characterized separately for each mode, just like the data input paths, as long as we correctly identify the conditions under which the control inputs are allowed to block each other and the data inputs. The benefit is that tighter delay estimates can now be obtained for the control inputs, leading to more accurate timing models.

The proposed method AdvChar uses any one of the propagation conditions that we identified as correct for this purpose. These conditions are viability [5], safe static [7], static co-sensitization [3], and the Brand-Iyengar condition [1]. For lack of space, only viability is considered here. Although the event propagation conditions under viability depend on the input arrival times which

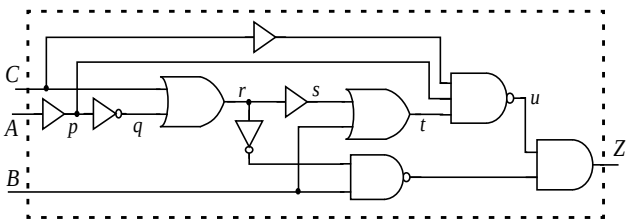


**Figure 1. Example circuit.**

are not known when a circuit is characterized, we show that the path delays obtained using viability are in fact guaranteed to be conservative, and the resulting timing models satisfy the correctness requirement. A similar result is available in the literature from [4] where the authors show that a combinational circuit can be conservatively characterized using viability. However, their method treats all the inputs uniformly, and places no restriction on the input combinations or vectors. In contrast, AdvChar takes advantage of the concept of control/data separation. In addition, as in ModeChar, it uses static sensitization for the data inputs, which is a tighter condition than viability, and hence results in tighter delay estimates.

AdvChar works as follows. Let COND denote the propagation condition being used (any one of the above mentioned conditions). Each mode is applied to the circuit, implying constants for certain control inputs. These constants are propagated as far as possible. If COND is an arrival-time dependent propagation condition such as viability, the arrival time for each constant is also calculated assuming that all the control inputs arrive at time 0 (this is arbitrary; any other arrival condition can be used). After constant propagation, the next step of AdvChar is to calculate the worst-case delay between each control input/output pair. As paths are traced during this process, the determination of whether a given path is blocked at a gate is made using COND. In this fashion, all control input/output pairs are processed, and the calculated delays are stored in the model along with the delays for the data input/output pairs. The following is a summary:

```
function AdvChar(CKT, M) {
/* Given: Circuit CKT, set of modes M */
/* Let C=control inputs, D=data inputs,
   O=outputs */
 for every mode m_i ∈ M {
  Apply m_i to CKT
  Propagate constants as far as possible
  Calculate D=>O delays with static sens.
  Calculate C=>O delays with COND
  store results in Timing Model
 }
 return TimingModel
}
```

The application of AdvChar to the combinational block in Fig. 1 yields the following model:

| Path | *C* = 0 | *C* = 1 |
|---|---|---|
| *A* => *Z:* | 6 | 3 |
| *B* => *Z:* | 2 | none |
| *C* => Z: | 4 | 5 |

Notice that under mode $C=0$, the path delay for $C \Rightarrow Z$ is 4, rather than 5. This is because path *C-r-s-t-u-Z* of delay 5 is false due to COND($t$,$u$)=0 under viability. This condition is false because input $t$ of gate $u$ has an unknown value while the top input has a controlling value and arrives earlier. Calculating a more accurate delay for path $C \Rightarrow Z$ can make a substantial impact on the output timing. Consider the output arrival time for mode $C=0$. Given the arrival times of 0, 0, 2 for inputs *A*, *B*, and *C*, respectively, the output arrival time would be calculated as 7 with the model from ModeChar, while it would be 6 (<7) with the above model, which is 14% lower. Depending on the specific arrival times and interaction with other blocks, the final delay impact can be much more significant.

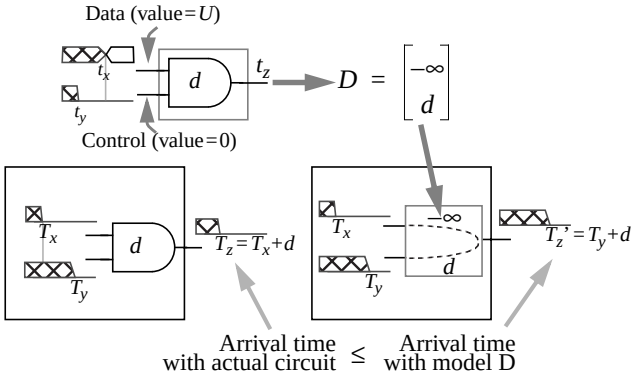


**Figure 2. Comparison of arrival times obtained with the actual circuit vs. timing model.**

To illustrate how path delays are conservatively calculated by AdvChar, we highlight one specific situation. Consider a single AND gate to be characterized. Assume that in one particular mode, input *y* of the gate is a control input, with logic value 0, and input *x* is data. Suppose that the input arrival times shown in Fig. 2 are used for characterization, so $t_y < t_x$. According to viability, path $x \Rightarrow z$ is false, while path $y \Rightarrow z$ is true, which results in the timing model *D* shown in Fig. 2. We now check whether this model gives a correct (conservative) estimate for the output arrival time for a specific instance of the block in comparison with the case where the actual gate (block) is used for timing analysis. For this specific instance, assume that the arrival times are $T_x$ and $T_y$, and that $T_x < T_y$. Further, the logic value of input *x*, which was *U* for characterization purposes, is actually 0. With the actual block, the output arrival time is calculated as $T_z = T_x + d$. On the other hand, using the timing model, we get $T_z' = T_y + d$. Though $T_z'$ is not identical to $T_z$, clearly $T_z' > T_z$ since $T_x < T_y$. Thus the timing model is indeed correct in that it does not lead to underestimation of arrival times.

Analysis of all possible cases leads to the conclusion that AdvChar always produces correct timing models. The reader is referred to [10] for a formal proof. An interesting observation is that the floating-mode condition, which is commonly used in the literature [2,3], does not produce correct timing models with AdvChar; this is demonstrated in [10].

## 3. CHARACTERIZATION of SEQUENTIAL BLOCKS

The previous section was mainly concerned with modeling of combinational blocks. Creating a timing model for sequential blocks is more difficult because of the need to capture setup/hold requirements for latches and flip-flops within the block. A straightforward way of applying AdvChar to sequential circuits is to create mode-dependent timing models for the combinational blocks between synchronizers and make all synchronizers visible at higher levels where they can be handled explicitly. This improves on the traditional *grey-box* modeling approach, which employs topological delay modeling on the combinational blocks between the synchronizers. In either case, the grey-box model has the disadvantage of exposing the circuit's structure and complexity to higher levels of analysis which is potentially inefficient as well as inapplicable in cases where no part of the intellectual property can be revealed. In contrast, our approach is a *black-box* type of modeling, based on a technique called *clock modeling* originally proposed in [6]. Through processing of the internal timing

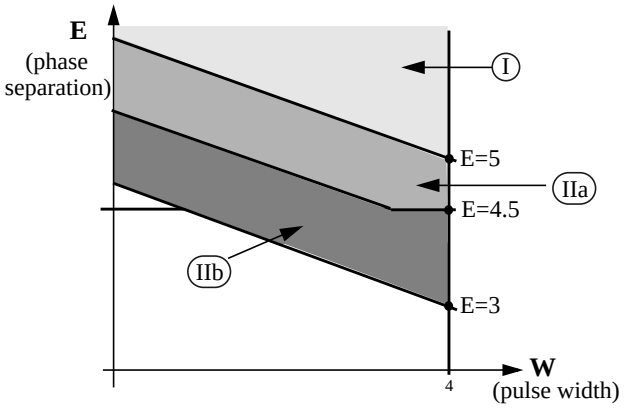


**Figure 3. Clock models obtained with topological analysis (I), and AdvChar (IIa and IIb).**

constraints, the clock modeling approach reduces all the timing constraints of a sequential block into a small set of constraints in terms of the circuit's clock parameters, those being clock pulse widths and phase separations. The resulting set of constraints is called the clock model.

The limitation of the method described in [6] is that it is based on topological delay analysis. Thus the clock models it produces can be pessimistic because of false paths. An improvement to this method is proposed in [9], which is based on ModeChar. The method of [9] produces a set of clock models, each being valid for a particular mode of circuit operation. However, the accuracy of this method is also limited because it cannot detect any false paths involving control inputs.

The extension of AdvChar to sequential circuits is as follows. For each circuit mode, constants are applied to the circuit and propagated as much as possible. Then the clock modeling technique is utilized to reduce the constraints. In this process, paths involving data inputs are processed using static sensitization, while those involving control inputs are processed using COND (any of the conditions mentioned before.) The reduced set of constraints obtained in this fashion constitute the clock model for this mode. The process is repeated for all circuit modes, resulting in a set of clock models. Since AdvChar can detect false paths from control inputs, the clock models it produces are more accurate than those found by the method of [9].

Fig. 3 illustrates how AdvChar can improve the accuracy of the clock model. Indicated with (I) is the clock model obtained with topological analysis. (IIa) and (IIb) indicate the clock models for two different modes. For both modes, the constraints are more relaxed and thus cover more combinations of the clock parameters. In other words, by eliminating false paths, AdvChar produces a more accurate timing model. Additional detail can be found in [10].

## 4. EXPERIMENTAL RESULTS

The AdvChar method has been implemented in a timing analysis tool and applied to the ISCAS-85 benchmark circuits as well as two industrial IP blocks. For the ISCAS-85 benchmarks, the mode information from [8] was used. The industrial IP blocks are a 32-bit ALU (alu32) and a 16-bit microprocessor (mpu). Both these IPs were characterized using 16 modes. Table 1 gives information about all the test circuits. We now compare the results of AdvChar with those of ModeChar and topological characterization to highlight the advantages of AdvChar.

**Table 1: Delay variation due to mode dependency.**

| Test circuit | Gate count | # of Modes | ModeChar | | AdvChar | |
|---|---|---|---|---|---|---|
| | | | Control | Data | Control | Data |
| c432 | 160 | 7 | 0% | 70% | 87% | 70% |
| c499 | 202 | 2 | 0% | 2% | 0% | 2% |
| c880 | 383 | 11 | 0% | 62% | 54% | 62% |
| c1355 | 546 | 2 | 0% | 10% | 0% | 10% |
| c1908 | 880 | 12 | 0% | 90% | 88% | 90% |
| c2670 | 1193 | 9 | 0% | 64% | 70% | 64% |
| c3540 | 1669 | 24 | 0% | 80% | 79% | 80% |
| c5315 | 2307 | 15 | 0% | 66% | 25% | 66% |
| c7552 | 3512 | 4 | 0% | 33% | 45% | 33% |
| alu32 | 817 | 16 | 0% | 98% | 93% | 98% |
| mpu | 2441 | 16 | 0% | 73% | 50% | 73% |

To demonstrate that the input-output delays of practical circuits can vary significantly across different modes, we calculated the delay variation for each test circuit. Here the delay variation of a circuit is defined as the maximum percentage difference in its delay between any two modes. The results are shown in Table 1. ModeChar calculates delay variation only for data inputs. For control inputs, it behaves the same as topological characterization, so there is no variation for control inputs with ModeChar. On the other hand, AdvChar calculates delay variation for both types of inputs. The amount of variation each test circuit exhibits depends on the circuit size, function and structure. As shown in the table, the delays of real circuits can vary by as much as 98%, depending on their mode. Those containing data processing logic with a high degree of control, such as ALUs, have larger delay variations, as in the case of alu32. Those that have few modes and little mode dependency (and hence few false paths due to control inputs), such as c499, have small delay variations, as expected.

The path delay variation for control inputs can be nearly as large as that for data inputs. For instance, the control inputs of c1908 have a delay variation of 88%, while the data inputs have 90% variation. The industrial circuit alu32 has 93% variation in its control input delays. Only AdvChar is able to calculate the path delays for control inputs accurately and capture their variations due to mode dependency.

To further illustrate the impact of mode dependency on timing, we found the maximum I/O delays for topological characterization, ModeChar and AdvChar. These are shown in Table 2 for all the test cases. As mentioned above, for circuits having little mode dependency, ModeChar and AdvChar cannot improve delay accuracy over topological characterization. But in other cases, they can make significant improvements. In Table 2, the cases for which ModeChar/AdvChar give better results are highlighted. The case of c5315 is noteworthy: while ModeChar is not able to improve upon the longest control path delay of 49, AdvChar produces a delay of 42, which is 14% lower. For the alu32 test case, AdvChar reduces the longest control path delay from 4.14 to 3.36, which is a 19% improvement.

**Table 2: Maximum I/O delays for the test circuits.**

| Test circuit | Topological | | ModeChar | | AdvChar | |
|---|---|---|---|---|---|---|
| | Cont | Data | Cont | Data | Cont | Data |
| c432 | 16 | 17 | 16 | 17 | 16 | 17 |
| c499 | 9 | 11 | 9 | 11 | 9 | 11 |
| c880 | 24 | 24 | 24 | 24 | 24 | 24 |
| c1355 | 15 | 23 | 15 | 23 | 15 | 23 |
| c1908 | 32 | 40 | 32 | 37 | 29 | 37 |
| c2670 | 32 | 30 | 32 | 30 | 32 | 30 |
| c3540 | 47 | 47 | 47 | 47 | 44 | 47 |
| c5315 | 49 | 47 | 49 | 40 | 42 | 40 |
| c7552 | 43 | 40 | 43 | 40 | 43 | 40 |
| alu32 | 4.14 | 8.74 | 4.14 | 8.72 | 3.36 | 8.72 |
| mpu | 6.57 | 8.90 | 6.57 | 8.26 | 5.93 | 8.26 |

The run-time efficiency of AdvChar is worth mentioning. For each mode, the run time of AdvChar is no more than that of topological characterization. Since the number of modes is usually small, and topological characterization is reasonably fast, AdvChar is well suited for processing large circuits. In our tests, the run time for AdvChar ranged from under a second (for c432) to approximately 15 minutes (for c3540 with 24 modes) on a Sun Ultra 10.

# 5. CONCLUSIONS

We presented a new timing characterization method that improves the accuracy of timing models over existing approaches by making efficient use of mode dependency. Large delay variations among different modes of circuit operation are captured for both control inputs and data inputs, eliminating the vast majority of false paths without resorting to expensive search-based or symbolic techniques to account for functionality. The method was also applied to the problem of clock modeling for sequential circuits. We demonstrated through experiments that our method can significantly enhance the accuracy of timing models, thereby minimizing performance penalties due to false paths.

# REFERENCES


[1] D. Brand, V. Iyengar, “Timing Analysis Using Functional Relationships,” *Proc. Int’l. Conf. on Computer Aided Design*, 1986, pp. 126-129.

[2] H.-C. Chen, and D.H.C. Du, “Path Sensitization in Critical Path Problem,” *IEEE Trans. on CAD*, vol. 12, Feb. 1993, pp. 196-207.

[3] S. Devadas, K. Keutzer, S. Malik, "Computation of Floating-mode Delay in Combinational Circuits: Theory and Algorithms," *IEEE Trans. on CAD*, vol. 12(12), Dec. 1993, pp. 1913-1923.

[4] Y. Kukimoto, R.K. Brayton, "Delay Characterization of Combinational Modules by Functional Arrival Time Analysis," *Int. Workshop on Timing Issues in the Spec. and Syn. of Digital Systems (TAU)*, 1999, pp. 151-156.

[5] P.C. McGeer, R.K. Brayton, *Integrating Functional and Temporal Domains in Logic Design: The False Path Problem and Its Implications*, Kluwer Academic Publishers, Boston, 1991.

[6] S.V. Venkatesh, R. Palermo, M. Mortazavi, K. Sakallah, “Timing Abstraction of Intellectual Property Blocks,” *Proc. Custom Integrated Circuits Conf.*, 1997, pp. 99-102.

[7] H. Yalcin, J. Hayes, "Event Propagation Conditions in Circuit Delay Computation," *ACM Trans. on Design Automation of Electronic Systems*, vol. 2 (3), July 1997, pp. 249-280.

[8] H. Yalcin, M. Mortazavi, R. Palermo, C. Bamji, K. Sakallah, “Functional Timing Analysis for IP Characterization,” *Proc. Design Automation Conf.*, 1999, pp. 731-736.

[9] H. Yalcin, M. Mortazavi, R. Palermo, C. Bamji, K. Sakallah, J. Hayes, “Fast and Accurate Timing Characterization Using Functional Information,” *IEEE Trans. on CAD*, vol. 20(2), Feb. 2001, pp. 315-331.

[10] htpp://www.eecs.umich.edu/acal/TimingAnalysis