Cyclic Timing Path Evaluation Using Commercial Static Timing Analysis Algorithms

Mackenzie J. Wibbels, Baudouin Chauviere, and Kenneth S. Stevens

Electrical and Computer Engineering Department, University of Utah

Abstract— **Many circuit design methodologies employ combinational logic with feedback to realize logic functions. Obtaining accurate delay values for cyclic timing paths is an impediment to the adoption of these technologies. Static timing analysis (STA) algorithms are the standard for digital timing validation but require that timing arcs form directed acyclic graphs. STA algorithms can be used to evaluate cyclic timing paths with a loss in accuracy by cutting cycles into acyclic segments and summing each segment in a cyclic timing path. This paper reports on sources of error in using static timing analysis to evaluate acyclic timing segments of cyclic timing paths, and shows that using timing segments to evaluate cyclic timing paths with STA can lead to both overly pessimistic and incorrect delay calculations. A method is presented to accurately evaluate segmented cyclic timing paths and detect drafting conditions using STA tools. The algorithm is executed on a number of common cyclic circuits and has been shown to accurately evaluate segmented cyclic path delays compared to fully unrolled circuit paths for different operating conditions, fanout loads, and timing path cuts.**

I. Introduction

Several digital circuit design methodologies employ combinational cycles to achieve power, performance, and area (PPA) benefits over acyclic counterparts. These circuits often use combinational feedback to implement sequential handshake elements. Handshake circuits create coupled oscillators that dictate the frequency of operation for such designs. Fig. 1 shows a post-layout system cyclic timing path between two instantiations of a handshake circuit module. Cyclic combinational circuits have also been employed to realize combinational functions at a reduced area and power relative to acyclic implementations [1].

Cyclic sequential logic design methodologies have gained little traction in commercial environments despite potential PPA benefits. One of the main impediments towards the adoption of these technologies is the lack of support for static timing analysis (STA) of cyclic timing paths by commercial electronic design automation (EDA) tools. STA algorithms use accurate models for delay, variation, and noise across multiple process corners and operating conditions. Additionally, variation and timing models used by STA algorithms continue to evolve in order to improve timing fidelity for advanced lithographic processes. For this reason, the cyclic path evaluation (CPE) flow presented in this paper leverages existing STA algorithms and tools. Leveraging STA tools allows the CPE flow to use modern STA models and algorithms, remain flexible to support future models and algorithms, and empower future adoption of cyclic sequential design methodologies by enabling timing verification using industry-standard STA tools.

STA algorithms evaluate circuit timing efficiently by transforming circuit structures into timing graphs. Circuit timing graphs are used to calculate timing topographically. Worst case arrival time and input slew are calculated from primary inputs and propagated through intermediate nodes to compute the best and worst output slew and delay. Topographical timing analysis requires the circuit timing paths to form a directed acyclic graph (DAG). DAGs are created for circuits with cyclic timing graphs by cutting timing arcs that form cycles in the design. However, the segmentation of cyclic timing paths causes increased variation and delay pessimism for advanced variation models and often leads to inaccurate evaluation of segmented timing paths.

Cyclic oscillating paths can also introduce STA delay error due to insufficient voltage stabilization between oscillations. Cyclic oscillating paths may cause gates to only partially transition to ground or *VDD* before the next input transition arrives. This phenomenon is called drafting and may require higher fidelity timing analysis which is out of the scope of STA [2].

This work makes the following contributions. 1) Reports STA inaccuracy introduced due to the segmentation of timing paths into acyclic timing segments 2) Presents a path segment characterization algorithm, which improves STA accuracy of decomposed path segments. 3) Presents a cycle stabilization algorithm which reports cyclic drafting conditions that may cause inaccurate STA delay calculations. 4) Develops a segment composition algorithm that composes characterized path segments to accurately evaluate the delay and variation of cyclic timing paths. To the best of the authors' knowledge, no method of directly handling variation of cyclic timing paths using STA algorithms has been previously developed. The algorithms have been developed in Tcl which allows the CPE to interface with commercial STA tools. Results generated in this paper use Synopsys PrimeTime and a 12nm technology node with Composite Current Source (CCS) delay models and Parametric On Chip Variation (POCV) models.

II. Background

A. Circuit Timing Evaluation

Two primary methods exist to evaluate circuit timing: simulation and static timing analysis. Simulation uses explicit

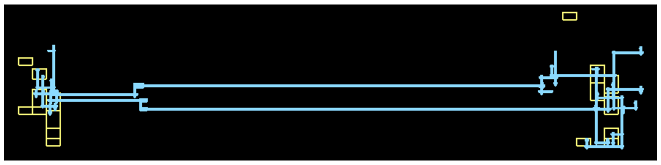

Fig. 1. A system level cyclic timing path created by handshaking between two instantiations of a handshake control circuit in a 12nm design. The standard cells (yellow) implementing the controller and the request / acknowledge wires (blue) between the controllers are highlighted.

input vectors and dynamic circuit computations to evaluate circuit performance. SPICE simulations provide high timing fidelity at the cost of run time. STA jointly optimizes run time and timing fidelity to provide efficient evaluation and scalability for large designs.

STA algorithms employ gate delay models to quickly and accurately compute path delays within a design. STA tools account for many second-order effects which contribute to circuit delay, including cross-talk and signal noise. Delay models use input slew and output capacitance to compute gate delay and output slew. Technology scaling has driven the advancement of timing models to provide better accuracy of non-linear device behavior for advanced process nodes.

Various methods exist to evaluate cyclic timing paths using both simulation and static timing analysis approaches. STA was chosen for this work because it is the de facto standard for commercial timing sign-off and the continued development of STA tools to support timing closure for advanced technology nodes.

B. Related Work

Integration of cyclic circuits into STA driven design flows is an area of ongoing research. Timing analysis methods can be separated into two categories. The first category uses custom timing analysis environments to evaluate and optimize circuit performance. These custom environments either rely on SPICE simulations or use custom timing environments to operate on data models used by STA tools [3], [4]. SPICE methods provide high timing fidelity at the cost of larger runtimes and also require a large number of simulations to evaluate design variation. Additionally, as shown in Fig. 1, cyclic timing paths often exist at a system level and therefore cannot be evaluated by running SPICE on a small hierarchical submodule.

Cyclic timing analysis methods that use STA data models within custom timing analysis engines have also been developed [2], [4], [5]. These methods promise improved scalability over SPICE based methods but require custom tools that must be maintained to handle advancements in delay and variation models. Timed separation of events (TSE) represents an analysis technique which computes minimum and maximum timing bounds between circuit pin transitions [3], [6]. Charlie diagrams represent another method of timing characterization that employs custom timing models, which express gate delay as a function of the time between consecutive input transitions [2], [7].

The primary method of evaluating cyclic timing paths using existing STA algorithms and tools requires cycle cutting of a cyclic timing graph. Several methods have been developed to cut and evaluate cyclic timing paths as acyclic segments [8], [9], [10]. The $set\_disable\_timing$ constraint can be used to disable timing propagation between $from\_pin$ and to_pin arguments to create a timing DAG to allow timing driven synthesis of cyclic sequential circuits. A method of evaluating cyclic timing paths which partitions paths into acyclic segments with overlapping portions has been developed [8]. Overlapping path segments allow for more accurate slew propagation at cut points but requires multiple STA runs to separately evaluate overlapping timing segments.

This presented CPE flow also relies upon cycle cutting methods to cut cyclic paths into acyclic segments, but presents a characterization algorithm to calculate accurate input slew values and path delays for acyclic path segments. A segment composition method is also developed, which derates variation of each timing segment across the complete composed timing path. Lastly, a method that detects drafting conditions and reports cyclic timing paths which may require higher fidelity timing analysis is presented. The flow has been applied to validate cyclic timing paths of a large 5nm chip.

III. Cycle Cut STA Path Evaluation Error

STA calculates path delay by propagating slew information across gates in the circuit timing graph. Cycle cutting introduces cuts to form an acyclic timing graph. Timing cuts may be applied using set_min_delay, set_max_delay, and $create\_clock$ constraints. Timing cuts can also be created by library cell definitions. Standard cell libraries cut timing arcs at the clock pins of sequential cells.

Timing cuts alter slew propagation and delay evaluation of STA tools by creating invalid slews. Invalid slews are slew values that are only used to calculate the delay of the first gate in a segment and are not used to calculate input slew values of downstream path gates. Instead, a zero delay slew transition is applied at the segment start point and is used to calculate the propagated input slews and subsequent segment gate delays for all gates excluding the initial gate. Invalid slews can be observed using the $report\_delay\_calculation$ command [11].

Path evaluation inaccuracy due to cycle cutting can be categorized into three sources: (1) variation error, (2) input slew (IS) error, and (3) input slew propagation (ISP) error.

Fig. 2 shows a circuit with a cyclic timing path from enA- to out- and a timing cut at pin G2/A2, which decomposes the cyclic timing path into three acyclic segments. Signal annotations {+,-} indicate rising and falling transition respectively. The circuit is mapped to a 12nm technology library with composite current source delay models and POCV variation models. The *set_disable_timing* command can be used to cut the cycle, but prevents timing arcs from starting or terminating at disabled timing endpoints. This paper instead uses set_min_delay and set_max_delay constraints in conjunction with *from_pin* and *to_pin* parameters to

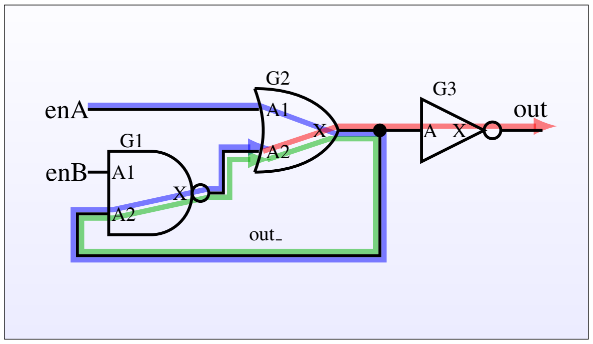

Fig. 2. Simple oscillating circuit with segmented timing arcs.

TABLE I
REPORTED MAXIMUM DELAYS OF RING OSCILLATOR SHOWN IN FIG. 2.

1 Conf.	2 Blue (ps / 3σ)	3 Green (ps / 3σ)	4 Red (ps / 3σ)	5 Nom. Dly. (ps)	6 Var. Dly. (ps)	7 Tot. IS Dly. (ps)	8 Tot. ISP Dly. (ps)
Unrl.	24.37 / 4.31	22.00 / 3.25	18.84 / 4.19	65.21	6.83	42.59	22.62
Cut	24.37 / 4.31	23.01 / 3.30	19.15 / 4.26	66.53	11.87	48.58	17.95

decompose cyclic timing paths into acyclic segments, which allows constraints to originate or terminate at cut gate pins.

Table I shows the effects of timing cuts on maximum delay path evaluation for the circuit in Fig. 2. Columns 2, 3, and 4 report the respective delay and variation for timing segments in the unrolled and cut versions of the circuit. In the cut version of the circuit the segment values are reported by STA directly. For the unrolled version, the values have been manually decomposed into separate nominal delay and 3σ variation segment values for comparison. Variation calculation and composition is discussed in greater detail in the following subsection. Column 5 reports the sum of the nominal segment delays. Column 6 reports the sum of the variation of the timing segments reported by STA in the cyclic and cut configurations of the circuit. From column 6 it can be seen that evaluation of cyclic path variation as the sum of decomposed timing segment variation leads to significant variation penalty.

As mentioned above, timing cuts produce two effects on path delay evaluation: (1) Pessimistic input slews are selected for the initial gate delay calculation and (2) zero delay slews are propagated to compute subsequent gate slews and delays. These effects can be seen in columns 7 and 8. Column 7 reports the invalid slew (IS) delay, which is the sum of the initial gate delays in the three segments. Column 8 reports the invalid slew propagation (ISP) delay, which is the sum of the subsequent gate delays in the three segments. For the max delay paths, the invalid slew selection increases delay estimates whereas invalid slew propagation decreases delay estimates.

A. Variation Error

Cycle cutting prevents accurate evaluation of cyclic path variation. Early variation models such as the on chip variation (OCV) model applied a fixed derate coefficient to each gate in a timing segment. Fixed derate models are overly pessimistic estimates for longer paths. Therefore advanced on chip variation (AOCV) was introduced to reduce pessimism of long paths [12]. AOCV uses a two dimensional lookup table indexed by path distance and depth to compute a gate derate value. The parametric on chip variation (POCV) model was introduced for process nodes below 90nm. POCV uses a coefficient to calculate the standard deviation of a gate's delay. Equations 1–3 show variation calculations for a path segment using POCV models. The standard deviation for each gate σ_{gate} in a path segment is calculated as the product of POCV coefficient (C_{pocv}) and the nominal gate delay ($Delay_{gate}$). The variance of the path segment σ^2_{path} is computed in Equation 2 by summing the variance of each individual gate. Equation 3 computes path variation by multiplying the effective standard deviation of the path with the $SigmaGuardband$ coefficient.

$$\sigma_{gate} = C_{pocv} * Delay_{gate} \tag{1}$$

$$\sigma^2_{path} = \sum_{i=1}^{n} {\sigma_{gate_i}}^2 \tag{2}$$

$$Variation_{path} = SigmaGuardband * \sqrt{\sigma^2_{path}} \tag{3}$$

The *SigmaGuardband* value shown in Equation 3 is defined by the user. The default value in PrimeTime is 3 sigmas. POCV liberty variation format (LVF) is an extension to the POCV format designed for process technologies 12nm and below. LVF extends POCV by adding $mean\_shift$ and $skewness$ parameters as well as output slew sensitivity based on input slews [13].

Cycle cutting path segmentation prevents variation values from being derated across full timing paths, which creates overly pessimistic results. One of the primary benefits of advanced and parametric variation models over OCV is improved derate values for long paths. POCV models gate variation as separate independent variables, which allows the standard deviation of a path to be computed from the sum of the variance of the path gates. Equation 4 shows that for n path segments of equal delay, the variation overhead due to segment composition grows as the square root of n. By applying Equation 4 to Fig. 2 which contains three composed timing segments ($n = 3$), we can see that $\sqrt{3} = 1.732$, which closely matches the difference reported "Var. Dly." column between the cyclic and unrolled rows in Table I.

$$\frac{\sum_{i=1}^{n}\sqrt{{\sigma_i}^2}}{\sqrt{\sum_{i=1}^{n}{\sigma_i}^2}} = \frac{n}{\sqrt{n}} = \sqrt{n} \tag{4}$$

Accurate evaluation of cyclic timing paths requires the variation of the full path to be calculated directly rather than adding segment variation values. Oscillating cyclic paths contain multiple correlated transitions through a single gate. Therefore, directly calculating cyclic path variation using POCV models and Equations 1–3 may also create overly optimistic variation results.

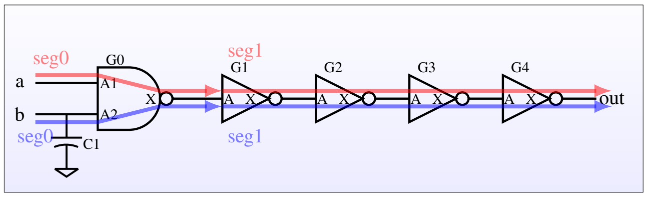

Fig. 3. Segmented timing path evaluation. Red arcs indicate max delay path constraints, and blue arcs indicate min delay path constraints.

TABLE II
REPORTED DELAY FOR TIMING PATH CONSTRAINTS SHOWN IN FIG. 3.

Path Type	G0/X+	G1/X-	G2/X+	G3/X-	G4/X+	Seg1 IS Dly.	Seg1 ISP Dly.	Seg1 Dly.	Tot. Dly.
Max	4.334	5.481	5.877	5.997	3.567	5.481	15.441	20.922	25.256
Cut Max	4.334	6.163	5.715	5.971	3.564	6.163	15.250	21.413	25.747
Min	6.279	6.115	5.958	6.010	3.568	6.115	15.536	21.651	27.930
Cut Min	6.279	5.426	5.870	5.996	3.567	5.426	15.433	20.859	27.138

B. Delay Error

Timing cuts produce invalid slews at cut locations. Fig. 3 shows an example invalid slew produced by applying timing constraints which begin and terminate at pin G1/A. The red arcs indicate max delay path constraints, and the blue arcs indicate min delay constraints. The paths covered by the blue arc have not been optimized to meet max delay requirements. A capacitor, $C1$, is added to model the unoptimized path and create a total capacitance on net b equal to the load capacitance of four unit-sized inverters. Table II shows the difference in path delay evaluation from $a-$ to $out+$ as a full path and as the composition of segments $seg0$ and $seg1$ with an invalid slew at pin G1/A. The timing cut at pin G1/A produces two effects: 1) The worst input slew is applied to pin G1/A to calculate the delay to G1/X, which is seen by the increased pessimism cut min and max delay G1/X- columns of Table II. 2) A zero delay slew is applied to G1/A to calculate the slew value at G2/A, which can be seen by the difference in delay in columns G2/X+ and the slews propagated to calculate delay at pins G3/X- and G4/X+.

1) Convergence of Invalid Slew Values: The error in gate delay due to invalid slew propagation decreases with path depth, as seen in Table II. Path slew convergence is shown in Fig. 4 for each gate in a path of inverters in a 12 nm finFET technology node. Maximum delay slew error is reported as a function of input slew applied to the path. The z axis reports the gate slew error for each gate, computed as the percent difference in slew values between gates in paths with the applied scaled input slew and the nominal 5.25ps slew value. The x axis is the "Path Index" that ranges from $[0..n-1]$ where n is the path length in gates. The error at index 0 is the slew error reported at the input of the first gate in the path. The y axis, "Scaled Path Input Slew", reports the scaled slew time applied at the input of the first gate in the path. The scaled path input slew value is achieved by applying the *set_annotated_transition* command to the input pin of the first gate in the path, with a nominal input slew time of 5.25ps scaled across the range of 52.5ns (10^4) to 52.5fs (10^{-2}).

As can be seen in Fig. 4, the largest slew error occurs

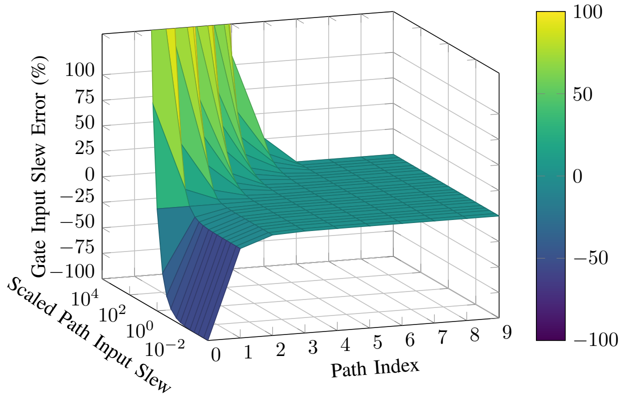

Fig. 4. Inverter gate input slew error as a function of path input slew and depth.

for gates near the start of the path and converges as the gate index increases. For a gate at index 1 with a -90% path input slew error, the gate output slew is approximately 10% faster. Gate delay sensitivity to input slew has been previously explored for 45nm design. The gate delay for a path consisting of nand gates with input slew times of 5ps and 50ps were shown to converge to 0.03% delay error at a path depth of 15 [14]. Fig. 4 shows that both fast and slow inverter slews converge as as the gate index increases. Fast slews experience RC filtering at the gate output network, where the minimum slew time is largely dictated by load capacitance and maximum drive current of the device. Slow input slews appear to speed up due to device gain. From experiments conducted on different devices, slow slew values appear converge more quickly for gates with greater small signal gain. Sensitivity experiments with different gate types have shown that input slew sensitivity is greater for simple gates with small output loads, whereas tests on complex gates which contain multiple diffusive stages, such as full adder circuits, show much lower slew sensitivity. This may be due to complex gates providing a higher effective small-signal gain due to multiple gain stages [15].

C. Drafting Condition Error

Drafting refers to a path transition condition in which gate outputs do not reach the full power or ground rail before the next path transition occurs. This affects the delay of subsequent oscillations [2]. Cyclic circuits present additional drafting challenges with internal loops, which may oscillate and cause drafting conditions. Significant research has been performed to provide high fidelity timing analysis of output slew and gate delay under drafting conditions [7]. However, drafting evaluation is not handled natively by STA algorithms. Drafting conditions can occur in oscillators with very short cycle times; therefore, a method of detecting and reporting cyclic timing paths which may exhibit drafting is presented.

D. Net Effect Cycle Cutting and Evaluation of Cyclic Paths

From the presented examples it can be seen that cycle cutting and invalid slew values produce three effects on the delay evaluation of segmented paths. First, path segmentation increases path variation pessimism due to variation calculated for individual path segments rather than a full path. Furthermore, cyclic paths contain multiple transitions on single gates, which may be correlated and are not accounted for in variation models. Second, invalid slews are created at path segment inputs. This produces two effects: 1) The selected invalid slew is pessimistic which increases pessimism for the first gate in the path. 2) Propagating the invalid zero delay slew value from the first gate in the path reduces path delay, which increases delay pessimism for min delay paths and decreases pessimism of max delay paths. For the path evaluated in the example, the invalid slew error is the dominant source of error which results in a pessimistic path delay evaluation. However, as will be shown later, there are instances where invalid slew propagation is the greatest source of delay error, which can cause overly optimistic max delay estimates, which can result in circuit errors.

IV. Cyclic Timing Path Evaluation

The CPE flow consists of three steps: (1) The path segment characterization (PSC) algorithm that characterizes delay and variation values of acyclic timing segments, which are created by decomposing full timing paths into segments. (2) The segment composition algorithm that reports the delay and variation of the full timing path from the composition of characterized segments. (3) The cycle stabilization algorithm reports cycles that may exhibit drafting conditions and require higher fidelity timing analysis.

A. Path Segment Characterization

The objective of the PSC is to characterize each cut point with a catalog of pairs of propagated slew values and fanin paths. Each cut point n_i is associated with a segment startpoint or primary output. The PSC characterizes slew values at cut points in the timing graph by propagating slew across multiple cut points. Slew values are propagated using the *set_annotated_transition* command. At each cut location, the algorithm maintains slew values for each fanin path segment.

Segments between connected cut locations in the timing graph may not be sufficiently long to accurately select a slew value during segment composition. Therefore, the path segment characterization algorithm records slew values produced by a sequence of fanin segments across multiple cut points. Each segment in the sequence fans in to the input of the next segment in the sequence. These sequences are referred to as *composite prefixes*.

The red path of Fig. 2 shows an example where the composite prefixes provides improved slew selection accuracy. For an output transition caused by the internal oscillating ring of the circuit, the max segment prefix depth of the green and blue prefix segments for G2/A2 have path depths of two, which may be insufficient to accurately select a characterized slew value for oscillating iterations of the loop. By applying composite prefixes, the PSC algorithm records slew values for different oscillations with composite prefixes consisting of the blue segment with zero or more green segments.

The PSC algorithm provides a target depth parameter *tpd* that defines the maximum gate length of the recorded composite prefix set. At each cut point a list of unique composite prefixes of length *tpd* is maintained. This allows the segment composition algorithm to select slew values during segment composition by matching composite prefix gates with the partially formed full timing path. In Fig. 2 for a *tpd* of 6, the segment characterization algorithm records three composite prefixes and transition pairs $\{\langle GGG, s_0\rangle, \langle BGG, s_1\rangle, \langle B, s_2\rangle\}$, where B denotes the blue segments and G denotes the green segment and s_i denotes the characterized input slew.

B. Segment Composition

Segment composition consists of two steps. (1) Segment prefix matching that selects and composes characterized path segments to form a complete timing path. (2) Segment variation composition that composes path segment delay and variation to evaluate a full timing path.

1) Segment Prefix Matching: Segment prefix matching iteratively selects characterized segments with prefixes with contain the greatest number of matching gate pins with the partially formed full timing path. The $tgtPrefxDepth$ parameter dictates the length of the longest recorded prefix and, therefore, the segment slew selection accuracy.

2) Segment Variation Composition: After characterized segments are selected, the segment variation composition algorithm evaluates the delay and variation of the full timing path. Adding the variation of each segment in a cyclic timing path does not leverage the path based derate values of current variation models, and produces overly pessimistic results. Evaluating multiple transitions on a single gate as independent gate transitions does not account for correlation between the transitions, which may also lead to overly optimistic results. Equation 5 shows the delay variance (σ^2) calculation for a cyclic path composed of signal transitions on nets a, b, and c. Notice the path contains two transitions on signal b.

$$\sigma^2_{path} = \sigma^2_{a+} + \sigma^2_{b+} + \sigma^2_{c-} + \sigma^2_{b-} \tag{5}$$

Both b+ and b- transitions occur on the same gate. These variation values may not be independent and therefore should not be added as independent variables. Therefore, during path composition, the correlation between transitions on the same gate must be taken into account. Equation 6 shows the calculation of variance for a linear combination of Gaussian random variables X and Y with covariance term σ_{xy}.

$$\sigma^2_{aX+bY} = a^2\sigma^2_x + b^2\sigma^2_y + 2ab\sigma_{xy} \tag{6}$$

Equation 7 calculates the correlation coefficient between two variable distributions such that $-1 \geq \rho_{xy} \leq 1$, where $\rho_{xy} = -1$ and $\rho_{xy} = 1$ represent fully anti-correlated and correlated variables respectively [16].

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{7}$$

Equation 8 is obtained by substituting Equation 6 into Equation 5. The covariance term $2\sigma_{b+-}$ of Equation 8 represents the correlated variation of the rising and falling transitions of signal b.

$$\sigma^2_{path} = \sigma^2_{a+} + \sigma^2_{b+} + \sigma^2_{c-} + \sigma^2_{b-} + 2\sigma_{b+-} \tag{8}$$

Substituting Equation 7 into Equation 8 yields Equation 9.

$$\sigma^2_{path} = \sigma^2_{a+} + \sigma^2_{b+} + \sigma^2_{c-} + \sigma^2_{b-} + 2\rho_{b_{+-}}\sigma_{b+}\sigma_{b-} \tag{9}$$

The transition polarity indexes of $\rho_{b_{+-}}$ in Equation 9 can be extended to specify input and output pin transition pairs and side input states by defining a mapping function from transition conditions, such as input pin, polarity, and side input states, to a specific correlation coefficient index. Equation 10 shows the calculation of the correlated variance of multiple transitions on a gate with gate pin pair indexes i and j.

$$CorrelatedVariance(\rho, \sigma_i, \sigma_j, i, j) = 2\rho_{ij}\sigma_i\sigma_j \tag{10}$$

An example matrix of correlation coefficient values is shown in Equation 11. The diagonal of the matrix is applied to slews with identical pin indexes. Therefore a cross-correlation value of $\rho_{ij} = 1$ is used, which represents fully correlated variables.

$$P = \begin{bmatrix} 1 & \cdots & \rho_{1n} \\ \vdots & \ddots & \\ \rho_{n1} & & 1 \end{bmatrix} \tag{11}$$

The correlation matrix P can be generated empirically using SPICE and Monte Carlo simulations, similar to standard cell variation models [17]. Correct variation models may also be obtained without additional library characterization by setting ρ_{ij} to the upper bound 1 for all i, j pairs. Calculations in this work apply a correlation matrix where equivalent transitions are fully correlated ($\rho_{ij} = 1$) for $i == j$, and transitions on the same gate are strongly correlated ($\rho_{ij} = 0.7$) for $i \neq j$. The value $\rho_{ij} = 0.7$ was selected as a typical correlation coefficient lower bound for strongly correlated variables. This result provides a method to calculate the delay variance of a cyclic path which accounts for the correlation of multiple transitions on the same gate.

V. Cyclic Timing Path Evaluation Flow Implementation

A. Path Segment Characterization

The path segment characterization algorithm operates on a constraint timing graph of the circuit (CTG). The constraint timing graph $CTG = \{N, E, T\}$ defines the timing graph for a set of segment timing constraints. CTG contains a set of timing segment end points $n_i \in N$ as nodes in the graph, a set of causal timing path segments $e_i \in E$ of edges, and a transition relation $t_i \in T$ where t_i is a tuple $\langle n_s \times e_i \times n_e \rangle$, where n_s is the first timing end point in e_i and n_e is the last.

Each edge of the CTG is a timing constraint applied to the circuit, and each node of the CTG is a primary I/O or a cut point in the timing graph which corresponds to a start or end point of a timing path constraint. Timing constraints used for CTG construction define the desired causal or evaluated path of the circuit. They are provided by the user and may be generated manually or by automatic characterization methods [18]. An example set of timing constraints for the circuit of Fig. 2 is shown in Table III.

TABLE III
TIMING CONSTRAINT SEGMENTS FOR THE CIRCUIT OF FIG. 2

Constraint Segment	Delay	Input Depth	Path Depth	Index
[enA- G2/A1- G2/X- G1/A2- G1/X+ G2/A2+]	max	6	2	1
[enB+ G1/A1+ G1/X- G2/A2-]	max	6	1	2
[G2/A2+ G2/X+ G1/A2+ G1/X- G2/A2-]	max	0	2	3
[G2/A2- G2/X- G1/A2- G1/X+ G2/A2+]	max	0	2	4
[G2/A2+ G2/X+ G3/A+ G3/X- out-]	max	0	2	5
[G2/A2- G2/X- G3/A- G3/X+ out+]	max	0	2	6

Segment characterization is performed as a search of the CTG. At each iteration of the search, fanout segments from a node $n_S \in T$ are used to propagate new output slew values to an adjacent node $n_e \in T$. Each instance of the search maintains a currently propagated slew value st_i for a composite prefix cp_i. The composite prefix cp_i is a sequence that contains the most recently explored timing segments, for which slew propagation results in a slew time st_i. During each iteration of the search an additional timing segment e_j is explored. A new slew time st_j is computed by applying st_i to the input of e_j. A new composite prefix cp_j is also generated by appending e_j to cp_i. To maintain the correct target prefix depth, cp_j is trimmed by removing the first, or least recently explored, segment while the remaining $cp_j[1:n]$ has a depth equal to or greater than tpd. The adjacent node n_e, composite prefix cp_j, and slew time st_j are added to the search if the node n_e has not been previously explored or if the node has been explored and one of two conditions hold. (1) The search arrives with a composite prefix cp_j that has been previously reported but computes a new slew value st_i that has not converged for cp_j. (2) The search arrives with a new composite prefix cp_j that has not been previously recorded for the node n_e. A slew st_i has converged with respect to previous slew st_j if st_i is less pessimistic than st_j or is within percentage convergence threshold of st_j.

The function $Gates(e_i, GT)$ maps each segment e_i to a sequence of gates $gt_i \in GT$ where gt_i is the gate sequence traversed by e_i. The function $DelayType(e_i) \mapsto \{Min, Max\}$ maps a segment path to a min or max constraint type.

The function $SegmentDepth(e_i, GT)$ returns the number of gates traversed by a timing segment in e_i. Function $OutputSlewTime(e_i)$ returns the calculated output slew for timing segment e_i. Output slew is calculated internally by the STA engine using path based analysis of a constraint edge e_i that maps from a node n_s to n_e (the transition $n_s \times e_i \times n_e$). The output slew calculated for node n_e depends on the slew annotated at n_s. If no slew value is annotated at n_s, the default path based analysis behavior of the STA tool is used to calculate the output slew time as described in Section III-B.

The cyclic characterization algorithm takes five inputs: the constraint timing graph CTG, a set of primary input nodes with slew values for the circuit IN, a gate sequence set GT, the timing prefix depth *tpd*, and the slew threshold *ct*. A search space $\{EX, PSD\}$ is maintained, where EX is a set of slew propagation nodes to explore and PSD is a prefix slew dictionary. Each propagation node $ex_i \in EX$ is a pair, $ex_i = \langle cp_i, s\rangle$, which consists of composite prefix cp_i and output slew s. Propagated composite prefixes and their associated slew values are recorded in the prefix slew dictionary PSD. Each prefix slew element $ps_i \in PSD$ maps a CTG node n_e to a prefix slew record $psr_i \in PSR$. Each psr_i is a pair $psr_i = \langle cp_i, st_o\rangle$ which contains a composite prefix cp_i and characterized slew time st_o such that $OutputSlew(cp_i) == st_o$, and st_o is the output slew computed by propagating an input slew time st_i across constraint segments $sg \in cp_i$. A composite prefix is a concatenation of prefix segments from EX and is formally defined by Definition 1.

Definition 1: A *Composite Prefix* cp_i is a sequence of timing path segments $[e_0 \dots e_n]$ such that $e_j \in E$, where $\forall e_j, e_{j+1} \in cp_i$ and $\exists t_j, t_{j+1} \in T$ such that n_e of t_j equals n_s of t_{j+1}.

The function $PrefixDepth()$ computes the depth of a composite prefix consisting of n timing segments as shown in Equation 12, where *InputDepth()* returns the target depth parameter *tpd* if the composite prefix e_0 begins at a primary input ($n_s \in IN$), and 0 otherwise. The $|Gates(e_i, GT)|$ term returns the number of gates in each path segment $e_i \in cp_i$.

$$PrefixDepth(cp_i, IN, GT, tpd) = \sum_{i=0}^{n-1} InputDepth(e_i, IN, tpd) + |Gates(e_i, GT)| \tag{12}$$

The $SlewConv(s_0, s_1, ct, delayType)$ function determines if slew s_1 has converged with respect to a previous slew s_0, a user specified threshold ct, and timing constraint type if either of the following conditions apply:

1) Max Delay: $s_1 \leq (s_0 * (1 + ct))$
2) Min Delay: $s_1 \geq (s_0 * (1 - ct))$

$PropagatePrefixSlew(cp_i, st_o, PSR, tpd) \mapsto \{T, F\}$ function defines the termination function for the slew propagation algorithm. For a composite prefix cp_i, output slew time st_o, previously slew prefix record PSR, and target prefix depth tpd, the search continues while no equivalent prefix has been recorded, the prefix is not contained in another prefix, and the slew has not converged as defined here.

1) $PrefixDepth(cp_i, IN, GT, tpd) \geq tpd$ and $\forall psr_j \in PSR : psr_i \neq psr_j$
2) $PrefixDepth(cp_i, IN, GT, tpd) < tpd$ and $\forall psr_j \in PSR : psr_i \not\sqsubseteq psr_j$
3) $\forall psr_j \in PSR$ where $cp_i == cp_j$ and $SlewConv(st_j, st_i, ct, CstrType(cp_i)) == F$

The $PropagatePrefixSlew$ function propagates slew values for composite prefixes of path depth greater than or equal to tpd which produce the most conservative path delay values. The third propagation condition, which propagates slew values when $SlewConv$ returns false, assumes that larger slew values produce greater path delays and smaller slew values produce smaller path delays. Although it is possible for large input slew times to produce negative gate delay values, which would invalidate this assumption, none of the propagated slew times produced negative gate delay values in the examples evaluated.

1) Implementation: Pseudocode for the prefix slew characterization algorithm is shown in Algorithm 1. The algorithm performs a depth first search of the constraint timing graph (CTG). Lines 2–8 initialize the prefix slew dictionary PSD and explore queue EX. For the CTG in Fig. 5, node G2/A2+ in PSD is initialized with composite prefixes $\{\{4\}, \{1\}\}$. Line 7 creates a new composite prefix slew record for the CTG node and appends the PSD entry.

The *GetFanoutPrefixes* function on line 28 returns composite prefixes for each newly explored CTG node. The function identifies fanout segments from the node (line 30). For each fanout segment, a new composite prefix is created by appending the segment to composite prefix *cp* (line 35). After the segment is appended, *cp* is trimmed by removing the first prefix segment while the *PrefixDepth* of the remaining composite prefix is greater than *tpd* (lines 36–37).

At each explored node, the algorithm evaluates the recorded slew. Termination of slew propagation is determined by the *PropagatePrefixSlew* function (line 19). *PropagatePrefixSlew* propagates a slew value when the search arrives at a previously explored CTG node which already contains an equivalent composite prefix, but propagates a new slew value *st* that has not converged within a specified threshold of the previously calculated slew value (line 25). Slew values are also propagated when the search arrives at a previously explored node with a composite prefix that was not previously recorded and is not a sub-sequence of a previously recorded composite prefix (line 27). If a composite prefix cp_j is shorter than *tpd* and is contained in another composite prefix cp_k ($cp_j \sqsubseteq cp_k$), the PSC algorithm marks cp_j as invalid (line 15). Invalid prefixes are removed

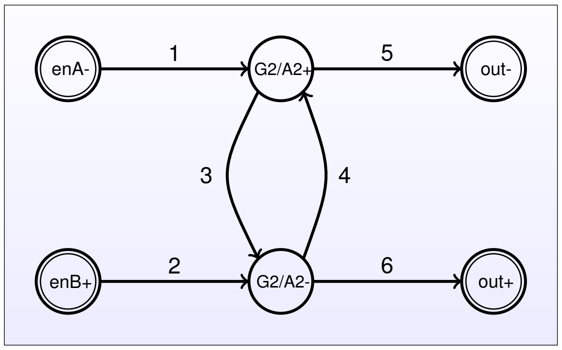

Fig. 5. Constraint timing graph of the circuit in Fig. 2 using timing constraint segments in Table III

Algorithm 1 Prefix Slew Characterization Algorithm

```
1: procedure CharacterizeTimingPaths($CTG, IN, GT, tpd, ct$)
2:     PSD = {}
3:     EX = {}
4:     for each $\langle n_s, e_i, n_e\rangle \in CTG$ do
5:         $st_o$ = OutputSlewTime($e_i$)
6:         $cp_i$ = CompositePrefix($e_i$)
7:         $PSD[n_e] = PSD[n_e] \cup \langle cp_i, st_o\rangle$
8:         $EX = EX \cup$ GetFanoutPrefixes($cp_i, st_o, tpd, CTG$)
9:     while $EX$ != {} do
10:        $\langle cp, st_o\rangle = removeFirst(EX)$
11:        $n_e$ = GetDestinationNode($cp[n], CTG$)
12:        $psr_i$ = PSD[$n_e$]
13:        if PropagatePrefixSlew($cp, st_o, psr_i, ct, tpd$) then
14:            $psr_i = psr_i \cup \langle cp, st_o\rangle$
15:            $invalidPrefixes$ = GetInvalidPrefixes($psr_i, tpd$)
16:            $PSD[n_e] = (psr_i - invalidPrefixes)$
17:            $EX = EX \cup$ GetFanoutPrefixes($cp, st_o, tpd, CTG$)
18:    return $PSD$;
19: procedure propagatePrefixSlew($cp, st, psr, ct, tpd$)
20:    isUniquePrefix = !($cp \in psr$)
21:    isSubsetPrefix = IsContainedSubSequence($cp, psr$)
22:    if !isUniquePrefix then            ▷ If equivalent composite prefix exists
23:        $\langle cp_i, st_i\rangle$ = GetRecordWithPrefix($cp, psr$)
24:        if !SlewConv($st_i, st, ct$, DelayType($cp[n]$)) then
25:            return $True$              ▷ Propagated slew has not Converged
26:    else
27:        return (!isSubsetPrefix) or ($cp$.depth $\geq tpd$)
28: procedure GetFanoutPrefixes($cp, st_i, tpd, CTG$)
29:    fanoutPrefixes = {}
30:    $n_e$ = GetDestinationNode($cp[n], CTG$)   ▷ Get node for last index of $cp$
31:    SetAnnotatedSlew($n_e, st_i$)
32:    update_timing –full                ▷ Compute STA with new slew values
33:    for each $e_j \in$ GetFanoutEdges($n_e$) do
34:        $st_o$ = GetOutputSlewTime($e_j$)
35:        $cp' = \{cp, e_j\}$
36:        while PrefixDepth($cp'[1:n]$) $> tpd$ do
37:            $cp' = cp'[1:n]$               ▷ Trim First element of $cp$
38:        fanoutPrefixes = fanoutPrefixes $\cup \langle cp', st_o\rangle$
39:    return fanoutPrefixes
```

from the prefix slew dictionary PSD and are not explored.

Table IV shows a prefix slew dictionary created by running the path segment characterization algorithm on the circuit in Fig. 2 that has a CTG shown in Fig. 5. The PSC algorithm was configured with a *tpd* of 6 and convergence threshold *ct* of 0.001 which specifies a slew convergence of 0.1%. Node G2/A2- is shown to have characterized prefix paths from both the primary inputs enB ($\{2\}$) and enA ($\{1,3\}$) and from the internal oscillation path ($\{3,4,3\}$). Although paths $\{2,4,3,4,3\}$ and $\{1,3,4,3\}$ are also unique composite prefixes for $tpd = 6$, both prefixes are trimmed to $\{3,4,3\}$ because the path has six gates.

2) Greedy Implementation: To improve run time, a greedy implementation of the slew propagation algorithm was developed. The greedy implementation concurrently propagates slews across multiple non-interfering prefix segments simultaneously. The greedy PSC algorithm replaces the prefix exploration queue EX with a data structure that allows items to be removed at a specific index. The algorithm greedily removes as many non-interfering segments as possible.

The function $Interfering(cp_i, cp_j) \mapsto \{T, F\}$ is true when the prefix segments cp_i and cp_j share the destination node n_e. If $Interfering(cp_i, cp_j)$ is false the two segments can be propagated concurrently with a single STA engine $update\_timing$ command.

At each search iteration, the greedy algorithm attempts to explore the largest possible set of non-interfering composite prefixes. The output slew time of each segment in the set is annotated and are all propagated using a single $update\_timing$ command. Composite Prefixes with propagated slew values are then pushed to the explore queue. The greedy implementation reduces the number of timing updates which is one of the most timing consuming steps of STA.

B. Segment Composition

The segment composition algorithm uses a prefix slew dictionary (PSD), variation correlation matrices for each gate transition, and a list of timing path segments that form the desired cyclic timing path. For each segment of the cyclic timing path, the segment composition algorithm matches gates in the full cyclic timing path with composite prefixes in the prefix slew dictionary. The characterized slew value of the prefix with the maximum matched gate count is used to calculate delay and variation values for the segment. Equation 2 is used to calculate the full path variance by summing gate variance values in each timing segment. Individual gate variance values are stored in a dictionary, where the index is the gate instance and the value is a pair consisting of the variance and the index in the transition correlation matrix for the gate type (P_{gate}). For each transition on the same gate, the matrix P_{gate} is indexed to report ρ_{ij} which is then applied in Equation 10 to calculate the covariance of the two transitions. The covariance of the transitions is added to the full path variance. After adding covariance of all transitions, path variation is calculated using Equation 3.

C. Cycle Stabilization Evaluation

The cycle stabilization evaluation (CSE) algorithm identifies cycles where drafting may occur. The stabilization of a cyclic path is evaluated as the difference between the sum of the gate output slew times and the sum of gate delays within a cycle. Unstable cycles are identified as cyclic paths which have a total gate output slew time that is greater than the total gate delay. The CSE algorithm allows the user to specify a guard-band margin for path stabilization using a *draftGB* parameter, which applies a fixed derate output slew time. The CSE algorithm is formulated as a negative cycle weight problem in which the objective of the algorithm is to identify cycles with a total weight that is less than zero. The algorithm operates on the same constraint timing graph as the path segment characterization algorithm. The weight of each edge e_i in CTG is computed as the difference between the sum of the gate delays in the path and the sum of slew times in a path created by applying input slew time st_i to n_s of e_i, as shown in Equation 13.

$$\begin{aligned} &StabilizationSlack(e_i, st_i, draftGB) = \\ &\quad TotalGateDelay(e_i, st_i) - \\ &\quad TotalOutputSlewTime(e_i, st_i) \times draftGB \end{aligned} \tag{13}$$

Negative weight cycles are identified using the Bellman Ford algorithm which reports the shortest path from a single

TABLE IV
MAX DELAY SLEW CHARACTERIZATION RESULTS FOR CTG OF FIG. 5

CTG Node	Characterized Prefixes
G2/A2-	$\langle\ \{2\}, 9.049\ \rangle$
	$\langle\ \{1,3\}, 7.639\ \rangle$
	$\langle\ \{2,4,3\}, 7.639\ \rangle$
	$\langle\ \{3,4,3\}, 7.163\ \rangle$
G2/A2+	$\langle\ \{1\}, 5.997\ \rangle$
	$\langle\ \{2,4\}, 5.990\ \rangle$
	$\langle\ \{1,3,4\}, 5.985\ \rangle$
	$\langle\ \{4,3,4\}, 5.985\ \rangle$
out-	$\langle\ \{1,5\}, 3.796\ \rangle$
	$\langle\ \{2,4,5\}, 3.796\ \rangle$
	$\langle\ \{3,4,5\}, 3.796\ \rangle$
out+	$\langle\ \{2,6\}, 4.406\ \rangle$
	$\langle\ \{1,3,6\}, 4.402\ \rangle$
	$\langle\ \{4,3,6\}, 4.402\ \rangle$

source node to all other connected vertices in the graph [19]. In order to detect all negative cycles in the graph, a virtual source node is created which connects to every node in the graph with zero weight edges.

The Bellman ford algorithm performs $(|N|-1)*|E|$ slack relaxation operation to refine the approximate minimum distance from the virtual source node to each node in the CTG. After completing $(|N|-1)*|E|$ iterations, the distances from the virtual source to nodes without negative cycles have converged. In order to detect negative cycles, the algorithm iterates over all edges in the CTG to determine if additional relaxations can be performed. A negative cycle exists if, for any node, an additional relaxation can be performed. If a negative cycle exists, the node and the necessary weight and predecessor information are returned in order to recreate the negative cycle.

VI. RESULTS

Table V and Table VI show the summary of 1200 circuit timing path evaluations of handshake controller modules which include a burst-mode controller (bmc), weak-condition half-buffer (wchb) [20], a micropipeline implemented using nand gates (mcrp), a speed independent micrpopeline implemented using complex logic gates (mcrpsi), and a mousetrap (mtrp). For each control circuit a cyclic timing path was created by composing two controller instances and evaluating the delay path from a primary input request through a full internal handshake cycle of the controller circuits to the primary output request of the controller pipeline. In order to evaluate the accuracy of the CPE and cycle cut (CC) STA, a golden STA model was created by unrolling the netlist of the cyclic timing path. For each evaluation the sensitivity of CPE and CC to slew load and number of path cuts was assessed by permuting simulation parameters across the following ranges: (1) the fanout of four (FO4) slew load at the cyclic path cut point was enumerated across $\{FO0, FO4, FO8\}$, (2) the number of timing cuts in the timing path was enumerated across $\{C0, C2, C4, CA\}$ where "A" indicates a cut at each gate input pin, and (3) the operating voltage across $\{$*0.3v, 0.6v*$\}$.

The columns of Table VI report length of the evaluated path in gates (2), the sum of the input slew time of gates in the path (3), the golden model variation of the unrolled path (4) and the golden model cyclic path delay (5).

Table V compares golden model data to the CC and CPE results and reports cycle stabilization evaluation (CSE) results. Column 2 reports the average target prefix depth required for the CPE flow to converge with the golden model results. Column 3 reports the number possible drafting condition instances identified across STA evaluations. The $*$ indicates that the number of stable cycles were not consistent for the varied number of timing cuts. Column 4 reports the average number of timing updates required for depth first search (DFS) and greedy implementation (GD) of the CPE flow. For large designs the number of timing updates primarily dictates STA runtime. Columns 5 and 6 report the percent difference of input slew time of gates in the path between GM vs. CC and GM vs. CPE respectively. Column 7 reports the percent of additional correlated variation calculated for the cyclic timing path using a correlation coefficient $\rho = 0.7$ for different polarity and pin transitions on a shared gate and $\rho = 1$ for multiple transitions on a shared gate with identical pins and transition polarity. Columns 8 and 9 compare uncorrelated path variation of the golden model and the two methods. Columns 10 and 11 report the percent delay difference between GM vs. CC and GM vs. CPE respectively.

As can be seen in Table V, the CPE algorithm maintains a consistent path slew propagation (6), variation (9), and delay (11) across all timing cuts and slew loads. There are instances in the nominal delay results where the lowest order bit does not match, these difference account for at most 30fs delay difference, which the authors believe is due to the precision of attribute reporting of the STA tool. Across all of the designs evaluated, the CC nominal maximum delay error (10) results range from -2.33% to $.17\%$ at 0.6V and -4.36% to $.49\%$ at 0.3V supplies. The delay ranges from negative to positive values indicates that CC STA may compute values above and below maximum full path delay estimates. Overestimates require greater delay margins; however, underestimates can produce circuit failures. The CC nominal minimum delay error results range from -4.72% to 0.03% at 0.6V and -6.36% to 0.01% at 0.3V. For minimum delay paths negative values indicate that default STA behavior will underestimate path delay which will add additional min delay margin, while positive values could produce circuit failures. The cycle stabilization algorithms reported unstable cycles in the *bmc* and *mcrp* module configurations, which both have relatively low handshake periods.

VII. CONCLUSION

The presented cyclic path evaluation algorithm provides a method to accurately evaluate cyclic timing path delay using STA tools. Sources of error in the evaluation of cyclic timing paths using STA tools to evaluate acyclic timing segments are identified. A prefix slew characterization algorithm is

TABLE V
COMPARISON OF CYCLE CUT (CC) FLOW, CPE FLOW AND GOLDEN MODEL TIMING PATH EVALUATION.

1 Design	2 Tgt. Prefix Depth	3 CSE Neg. Cycles (CC/CPE)	4 Number Updt. CPE (DFS/GD)	5 Slew Difference CC vs. GM (Min/Avg./Max)	6 Slew Difference CPE vs. GM (Min/Avg./Max)	7 Corr. Var. (%)	8 Variation Diff. CC vs. GM (Min/Avg./Max)	9 Variation Diff. CPE vs. GM (Min/Avg./Max)	10 Delay Diff. CC vs. GM (Min/Avg./Max)	11 Delay Diff. CPE vs. GM (Min/Avg./Max)
Max Delay 0.3v:										
bmc	6.89	2*/9	36.89/7.67	-11.59/-7.43/-1.79	0.00/ 0.00/0.00	57.14	93.26/172.06/268.92	0.00/0.00/0.00	-4.36/-2.71/-0.67	0.00/0.00/0.00
wchb	3.33	0/0	28.22/3.89	-7.28/-2.06/ 0.00	0.00/ 0.00/0.00	74.59	98.56/188.00/290.62	0.00/0.00/0.00	-1.35/-0.44/ 0.02	0.00/0.00/0.00
mtrp	2.56	0/0	24.00/2.89	-1.33/-0.51/ 0.00	0.00/ 0.00/0.00	72.29	91.44/165.03/236.04	0.00/0.00/0.00	-0.25/-0.09/ 0.00	0.00/0.00/0.00
mcrpsi	3.22	0/0	30.22/3.44	-1.96/-1.45/-0.79	0.00/ 0.00/0.00	73.91	88.72/188.58/319.53	0.00/0.00/0.00	-0.23/ 0.09/ 0.49	0.00/0.00/0.00
mcrp	3.67	6*/9	32.11/4.33	-8.50/-4.05/-0.34	0.00/ 0.00/0.00	66.39	92.31/189.81/311.45	0.00/0.00/0.00	-2.39/-1.07/-0.08	0.00/0.00/0.00
Min Delay 0.3v:										
bmc	6.89	7*/6	36.89/7.67	-2.22/-0.83/ 0.00	0.00/ 0.00/0.00	57.18	96.65/173.15/263.22	0.00/0.00/0.00	-6.36/-2.22/-0.01	0.00/0.00/0.00
wchb	3.33	0/0	28.22/3.89	-0.54/-0.11/ 0.13	0.00/ 0.00/0.00	75.24	98.91/187.68/288.13	0.00/0.00/0.00	-2.32/-0.72/ 0.01	0.00/0.00/0.00
mtrp	2.56	0/0	24.00/2.89	-0.02/-0.01/ 0.00	0.00/ 0.00/0.00	74.86	95.04/171.98/248.36	0.00/0.00/0.00	-0.59/-0.21/ 0.00	0.00/0.00/0.00
mcrpsi	3.22	0/0	30.22/3.44	-0.24/-0.07/ 0.00	0.00/ 0.00/0.00	78.18	94.81/196.34/329.83	0.00/0.00/0.00	-1.44/-0.47/0.00	0.00/0.00/0.00
mcrp	3.67	9/9	32.11/4.33	-1.14/-0.37/ 0.00	0.00/ 0.00/0.00	70.25	95.40/194.13/315.91	0.00/0.00/0.00	-4.85/-1.61/-0.01	0.00/0.00/0.00
Max Delay 0.6v:										
bmc	6.33	9/9	35.33/7.00	-9.07/-6.15/-1.28	0.00/ 0.00/0.00	59.55	93.85/173.65/272.26	0.00/0.00/0.00	-2.33/-1.57/-0.33	0.00/0.00/0.00
wchb	2.67	0/0	25.33/3.67	-6.59/-1.57/ 0.00	0.00/ 0.00/0.00	74.06	97.28/187.39/292.59	0.01/0.00/0.00	-0.68/-0.17/ 0.05	-0.01/0.00/0.00
mtrp	2.44	0/0	23.67/2.56	-1.18/-0.45/ 0.00	0.00/ 0.00/0.00	76.68	96.98/175.45/253.81	0.00/0.00/0.00	-0.14/-0.05/ 0.00	0.00/0.00/0.00
mcrpsi	2.44	0/0	27.00/2.89	-1.72/-1.22/-0.60	0.00/ 0.00/0.00	81.65	98.40/200.61/334.91	0.00/0.00/0.00	-0.25/-0.05/0.17	0.00/0.00/0.00
mcrp	3.56	9/9	31.56/4.11	-7.96/-3.00/ 0.00	0.00/ 0.00/0.00	68.49	97.26/192.73/334.79	0.01/0.00/0.00	-1.74/-0.54/ 0.01	0.00/0.00/0.00
Min Delay 0.6v:										
bmc	6.33	9/9	35.33/7.00	-2.02/-0.79/-0.01	0.00/ 0.00/0.02	58.98	95.52/169.46/257.21	0.00/0.00/0.01	-4.72/-1.80/-0.01	0.00/0.00/0.01
wchb	2.67	0/0	25.33/3.67	-0.55/-0.07/ 0.11	0.00/ 0.00/0.00	74.54	97.61/186.57/289.16	0.00/0.00/0.00	-1.53/-0.46/ 0.00	0.00/0.00/0.00
mtrp	2.44	0/0	23.67/2.56	-0.02/-0.01/ 0.00	0.00/ 0.00/0.00	76.30	96.78/174.22/252.23	0.00/0.00/0.00	-0.46/-0.17/ 0.00	0.00/0.00/0.00
mcrpsi	2.44	0/0	27.00/2.89	-0.03/ 0.04/ 0.23	-0.04/-0.01/0.00	82.61	99.77/201.75/336.65	0.00/0.00/0.00	-0.96/-0.31/0.00	0.00/0.00/0.00
mcrp	3.56	9/9	31.56/4.11	-0.96/-0.29/ 0.20	0.00/ 0.00/0.00	69.56	97.54/193.28/328.32	0.00/0.00/0.00	-3.52/-1.11/0.03	0.00/0.00/0.00

TABLE VI
GOLDEN MODEL RESULTS.

1 Design	2 Num. Gates Path	3 Golden Model slew (ps) (Min/Avg./Max)	4 Golden Model Variation (ps) (Min/Avg./Max)	5 Golden Model delay (ps) (Min/Avg./Max)
Max Delay 0.3v:				
bmc	19	1254.70/1520.91/1749.88	140.03/167.52/191.59	1205.13/1623.28/1623.28
wchb	19	650.06/ 876.74/1065.56	124.36/149.60/171.64	1172.24/1524.92/1524.92
mtrp	14	995.42/1120.21/1244.89	269.96/286.51/303.10	1766.09/2003.91/2003.91
mcrpsi	22	854.09/1025.10/1198.36	208.80/234.76/260.87	1679.80/1966.21/1966.21
mcrp	22	1483.13/1632.68/1790.19	124.49/136.67/149.96	1406.90/1633.15/1633.15
Min Delay 0.3v:				
bmc	19	1006.19/1255.73/1475.74	89.74/107.60/123.83	1005.07/1391.05/1391.05
wchb	19	640.06/ 852.83/1034.87	92.72/109.77/124.94	1125.57/1460.21/1460.21
mtrp	14	979.59/1093.13/1211.81	179.56/190.23/201.36	1706.16/1930.62/1930.62
mcrpsi	22	717.47/ 890.39/1064.72	127.23/145.24/163.52	1448.47/1741.00/1741.00
mcrp	22	1483.13/1610.56/1756.09	98.48/105.83/114.50	1406.91/1610.26/1610.26
Max Delay 0.6v:				
bmc	19	282.81/ 353.53/ 413.10	11.02/ 13.77/ 16.19	241.60/ 335.64/ 335.64
wchb	19	152.19/ 212.92/ 263.45	10.00/ 12.52/ 14.79	251.58/ 335.82/ 335.82
mtrp	14	231.08/ 271.83/ 311.69	21.62/ 23.61/ 25.54	379.14/ 445.19/ 445.19
mcrpsi	22	238.34/ 290.03/ 342.02	18.59/ 21.07/ 23.57	396.73/ 473.50/ 473.50
mcrp	22	300.11/ 351.45/ 401.83	8.89/ 10.95/ 13.08	272.75/ 336.87/ 336.87
Min Delay 0.6v:				
bmc	19	237.27/ 303.12/ 360.80	8.30/ 10.43/ 12.40	210.36/ 296.82/ 296.82
wchb	19	150.19/ 206.63/ 255.08	8.76/ 10.73/ 12.56	247.40/ 326.56/ 326.56
mtrp	14	227.65/ 262.10/ 298.47	18.02/ 19.50/ 21.04	364.81/ 423.63/ 423.63
mcrpsi	22	189.57/ 241.44/ 293.98	13.17/ 15.30/ 17.48	330.95/ 409.35/ 409.35
mcrp	22	300.11/ 342.99/ 389.24	7.63/ 9.08/ 10.73	272.75/ 329.27/ 329.27

developed, which reduces sources of STA error by characterizing prefix paths to calculate accurate slew values at timing segment cuts.

The cycle stabilization algorithm presents a method to report paths where STA results may be invalidated due to cycle drafting conditions. The cycle stabilization algorithm reported potential drafting conditions for the *bmc* and *mcrp* designs in module configurations without inserted buffer or delay elements.

The segment composition algorithm presents a method to combine segment variation, which leverages path based POCV derate across multiple timing segments and accounts for covariance between multiple transitions on a gate. In certain configurations the lack of slew characterization led STA tools to underestimate max delay segments, which can result in unmet timing constraints and result in incorrect circuit behavior. The CPE flow is applied to several handshake controller circuits, and results show that slew propagation and segment variation composition produces results equivalent to the golden model.

Without using the CPE flow, maximum nominal delay errors range from -4.72% to 0.49% and variation error ranges from 88.72% to 334.91%. Nominal minimum delay errors range from -6.36% to 0.03% and variation error range from 94.81% to 336.65%. The greatest reported source of pessimism is variation error, particularly at the near threshold 0.3V operating condition, where 3σ variation adds significant path delay overhead.

The cyclic timing path evaluation flow has been shown to accurately evaluate cyclic timing paths using STA tools for different operating conditions, fanout loads, and for an arbitrary number of timing cuts. The cyclic path evaluation flow provides a step towards commercial STA support for design methodologies that employ cyclic timing paths.

ACKNOWLEDGMENTS

This material is based upon work supported by the Granite Mountain Technologies, Inc. The authors would like to thank Daniel R. McCulley for his contributions to this work.

References

[1] M. D. Riedel, "Cyclic Combinational Circuits," Ph.D. dissertation, California Institute of Technology, 2004.

[2] A. Winstanley and M. Greenstreet, "Temporal Properties of Self-Timed Rings," in *Correct Hardware Design and Verification Methods*, ser. Lecture Notes in Computer Science. Springer, Aug 2001, pp. 140–154.

[3] S. M. Burns, "Performance Analysis and Optimization of Asynchronous Circuits," Ph.D. dissertation, California Institute of Technology, USA, 1992.

[4] N. Xiromeritis, S. Simoglou, C. Sotiriou, and N. Sketopoulos, "Graph-Based STA for Asynchronous Controllers," in *29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, July 2019, pp. 9–16.

[5] W. Hua, Y.-S. Lu, K. Pingali, and R. Manohar, "Cyclone: A Static Timing and Power Engine for Asynchronous Circuits," in *26th International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE, May 2020, pp. 11–19.

[6] H. Hulgaard, S. M. Burns, T. Amon, and G. Borriello, "An Algorithm for Exact Bounds on the Time Separation of Events in Concurrent Systems," *IEEE Transactions on Computers*, vol. 44, pp. 1306–1317, nov 1995.

[7] J. C. Ebergen, S. Fairbanks, and I. E. Sutherland, "Predicting Performance of Micropipelines Using Charlie Diagrams," in *Proceedings of the Fourth International Symposium on Adanced Research in Asynchronous Circuits and Systems*. IEEE, March 1998, pp. 238–246.

[8] W. Lee, V. S. Vij, and K. S. Stevens, "Timing Path Driven Cycle Cutting for Sequential Controllers," *The ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21,4, no. 64, pp. 1–25, Sept 2016.

[9] G. Gimenez, A. Cherkaoui, G. Cogniard, and L. Fesquet, "Static Timing Analysis of Asynchronous Bundled-Data Circuits," in *24th International Symposium on Asynchronous Circuits and Systems*. IEEE, May 2018, pp. 110–118.

[10] W. Lee, T. Sharma, and K. S. Stevens, "Path Based Timing Validation for Timed Asynchronous Design," in *The 29th International Conference on VLSI Design (VLSID)*. IEEE, Jan 2016, pp. 511–516.

[11] Synopsys, *PrimeTime User Guide and Reference Manual*, Synopsys, Inc., 690 East Middlefield Road, Mountain View, CA 94043 USA, 2017.

[12] S. Walia, "PrimeTime Advanced OCV Technology," Synopsys, Inc., White Paper, April 2009.

[13] Synopsys, *Liberty User Guide and Reference Manual*, Synopsys, Inc., 690 East Middlefield Road, Mountain View, CA 94043 USA, 2017.

[14] T.-B. Chan, P. Gupta, A. B. Kahng, and L. Lai, "DDRO: A Novel Performance Monitoring Methodology Based on Design-Dependent Ring Oscillators," in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, March 2012, pp. 633–640.

[15] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, 5th ed. John Wiley & Sons, January 2009.

[16] D. Wackerly, W. Mendenhall, and R. L. Scheaffer, *Mathematical Statistics with Applications*, 7th ed. Belmont, CA: Thomson Brooks/Cole, January 2008.

[17] S. Mittal, "A Survey of Architectural Techniques for Managing Process Variation," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 1–29, May 2016.

[18] M. J. Wibbels and K. S. Stevens, "Causal Path Identification for Timed and Sequential Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 3, pp. 571–582, March 2022.

[19] R. Bellman, "On a Routing Problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[20] A. M. Lines, "Pipelined Asynchronous Circuits," Master's thesis, California Institute of Technology, Pasadena, CA, 1998.