Path Based Timing Validation for Timed Asynchronous Design

William Lee Tannu Sharma Kenneth S. Stevens
Electrical & Computer Engineering, University of Utah
Email: *william.lee@utah.edu, tannu.sharma@utah.edu, kstevens@ece.utah.edu*

***Abstract*—Timing is an important parameter necessary to ensure the correctness of a design. Timed asynchronous designs can have complex timing paths that include combinational cycles. Commercial electronic design automation (EDA) tools do not support asynchronous designs because timing graphs are required to be acyclic. This paper reports on a methodology that enables commercial tools to support full cyclic path timing validation of timed asynchronous designs.**

I. Introduction

Scaling has revolutionized circuit design, allowing designs to be produced with over four billion transistors on a single chip. This capability is enabled with commercial electronic design automation (EDA) tools which are continuously enhanced to handle more complex circuits and technology nodes. With the increasing complexity of circuit design, obeying timing is a challenge. Commercial EDA tools offer well developed static timing analysis (STA) algorithms to validate timing constraints on clocked systems represented as a directed acyclic graph (DAG) [1].

The advent of novel circuit design methodologies, like asynchronous circuits, can enable a circuit to operate at multiple frequencies with power and performance benefits. Numerous advantages are accrued through employing commercial EDA tools for asynchronous design approaches. These tools have support for leading technology features such as double patterning and multiple timing corners, and design productivity is enhanced. Unfortunately asynchronous circuit designs can not be directly supported with commercial EDA tools.

The lack of commercial EDA support is largely due to the disconnect in the timing models employed for clocked and asynchronous design. Numerous challenges must be overcome to support asynchronous timing models in commercial EDA tools. Timing paths in asynchronous design are not simple combinational paths, as is the case with clocked design. Many timing paths in asynchronous design must be calculated based on the specific logic being employed. Such timing constraints often consist of two or more related paths. Nearly all asynchronous designs contain combinational cycles, which also must be evaluated. Asynchronous sequential controllers often have hazards that can be avoided if specific delay relationships hold. Timing constraints that make hazards unreachable must hold for design correctness. Other timing constraints exist in timed asynchronous circuits to optimize and validate performance. Most of the timing constraints to ensure circuit performance are cyclical.

These challenges need to be resolved for asynchronous design to employ commercial EDA tools. This paper specifically addresses the challenge of creating accurate path based delays for asynchronous sequential circuits, including paths that contain cycles. A methodology is developed that utilizes the static timing analysis (STA) algorithms used by commercial EDA tools.

II. Background

Timing in a system is employed to enforce specific event sequencing in a design. This is normally enforced with a clock signal, where the cycle time must be greater than the combinational propagation delay between pipeline stages. Timing is where asynchronous design differs from the clocked designs. Bundled data asynchronous design employs similar timing requirements, but timing is localized, flexible, and irregular. This creates problems and challenges for circuit optimization and validation, but also can provide power and performance advantages.

A. Asynchronous Designs

Asynchronous designs communicate and synchronize based on local handshake signals which identify data validity and the ability to accept new data transactions. The communication link is called a *handshake channel* that consists of data wires, a request signal (req) identifying data validity, and an acknowledgment signal (ack) indicating the data has been accepted [2] [3].

Performance evaluation for asynchronous designs is inherently different than that for clocked design because each handshake channel implements a silicon oscillator designed to operate at a particular frequency that matches the delay of the associated data path. The handshake channel implements the oscillator as a timing cycle. The timing path is even more complicated in four-cycle protocols since each data transfer consists of both rising and falling transitions on req and ack with these signals propagating along many if not all the

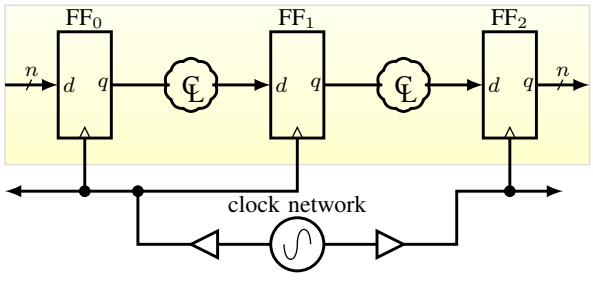

Fig. 1. Clocked design. Frequency and datapath delay of first pipeline stage is constrained by $FF_i/clk\uparrow_j \mapsto FF_{i+1}/d + margin \prec FF_{i+1}/clk\uparrow_{j+1}$

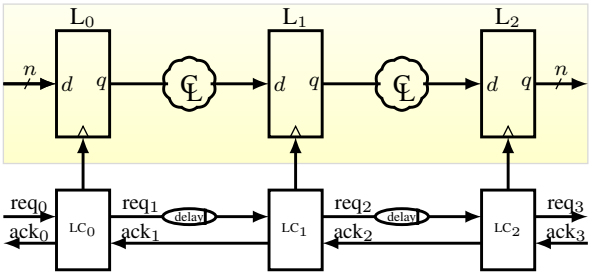

Fig. 2. Timed (bundled data) handshake design. Delay sized by RT constraint $req_i\uparrow \mapsto L_{i+1}/d + margin \prec L_{i+1}/clk\uparrow$. Each $req_i\uparrow$ handshake on LC_i indicates new data is presented to pin d of L_i.

gates in the handshake cycle. Commercial CAD does not support combinational cycles, and so external means must be implemented to support such cyclic timing paths.

Many asynchronous design styles, such as bundled data, require timing constraints to hold for the circuit to operate correctly. The LC blocks in Fig. 2 implement the silicon oscillator that control the frequency of operation for a pipeline stage, the synchronization between pipeline stages, and generate a clock signal to store data in pipeline latches. These are implemented as asynchronous finite state machines (AFSM). Here, we assume these controllers are implemented with combinational logic where state holding logic contains combinational cycles. These AFSMs often have hazards which must be made unreachable by controlling the delays in the circuit. The hazards are commonly associated with the combinational cycles that implement the state holding function.

This design approach poses multiple requirements which are not directly supported by the commercial EDA tool flow. First, timing paths must be explicitly identified that relate to the specific AFSM used in the design. Second, the timing graph of the circuit must be represented as a DAG. Third, cyclical timing paths must be evaluated based on a DAG timing graph representation.

In general, all the timing paths required to evaluate a sequential circuit can not be known in single iteration if the timing graph is represented as a DAG. Also, a sequential circuit represented as a DAG will have some of the timing paths cut [4]. Thus, complete timing path analysis requires multiple timing path partitions, multiple acyclic timing graph representations, and multiple STA runs.

B. Relative Timing

Relative timing (RT) defines a timing relation between two timing paths that start with a common point of divergence (pod) and ends at two distinct points of convergence (poc) $\mathsf{pod} \mapsto \mathsf{poc}_0 + m \prec \mathsf{poc}_1$ [5]. The maximum delay (max-delay) from pod to poc_0 plus a margin must be less than the minimum delay (min-delay) from the pod to poc_1.

Fig. 2 shows a simple linear bundled data pipeline, identifying the timing constraints between the control path and the data path. This timing constraint is similar to the one in synchronous circuits, which is shown in Fig. 1. Data L_{i+1}/d must arrive before the clock L_{i+1}/clk in order to store data correctly. The RT methodology can represent timing in a combinational or sequential circuit with combinational feedback loops. RT timing model is applied throughout this work because of its generality across timed asynchronous designs.

The timing margin (RT slack) for the RT constraint is the amount of time difference between the two timing paths. The RT slack is calculated by subtracting the delay value of the maximum delay path $\mathsf{pod} \mapsto \mathsf{poc}_1$ from the minimum delay path $\mathsf{pod} \mapsto \mathsf{poc}_0$.

C. Controller Indexing for Mapping Timing Constraints

In our approach, asynchronous sequential controllers become the focal point for timing validation, much like the registers are in a clocked design. Every AFSM controller has been characterized with a set of RT constraints which must hold for its correct operation. The source of all timing paths to be validated are RT pods, which are all located inside the characterized controllers. The ending points of the paths are RT pocs which may be inside or outside of the controller.

Since timing evaluation is performed local to each controller, a representation is constructed to specify external module location relative to the current controller. The controller under evaluation is identified with an instance label "$i1". Upstream controllers (accessed through the "ack" port in Fig. 2) are referenced as "$i0" design blocks; downstream pipeline elements (accessed through the "req" port) are referenced as "$i2" macros. Registers are connected to the controller through the "clk" output port and are identified by appending a capital R to the label (e.g. $i0R, $i1R, $i2R).

III. Relative Timing Verification

The relative timing verification flow used in this work is shown in Fig. 3. This flow enables commercial EDA tools to be used with sequential controllers and asynchronous circuit design. Fig. 4 is an example circuit used to explain the flow.

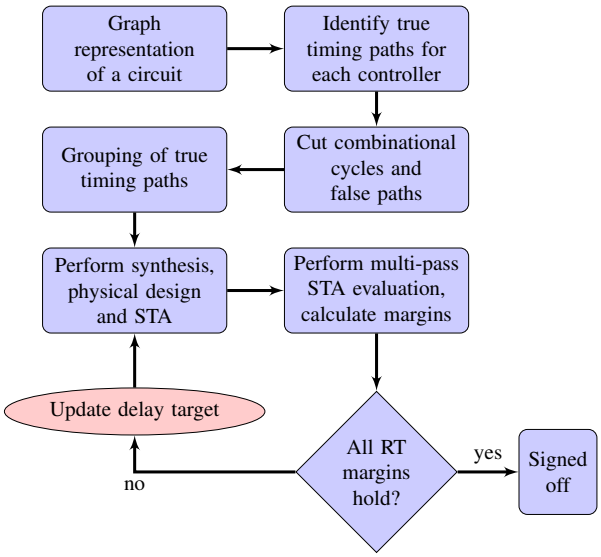

Fig. 3. Relative timing verification flowchart.

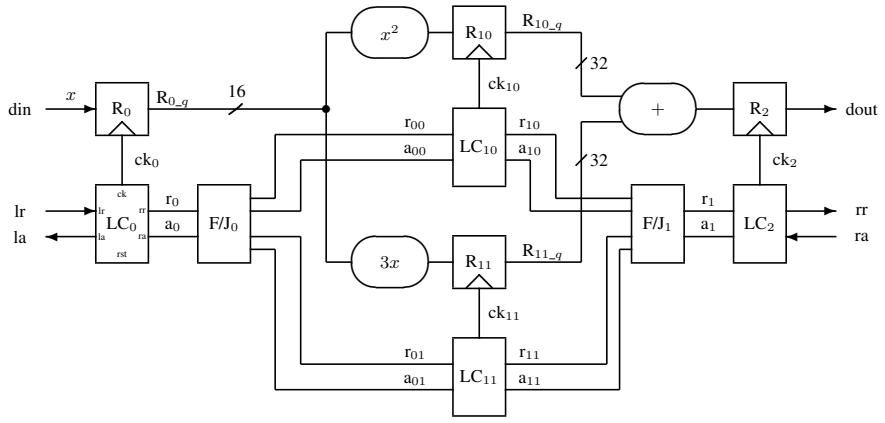

Fig. 4. Example design: a simple ASIC mathematical pipeline segment computing dout $= x^2 + 3x$ [6]

Relative timing constraints specify timing paths by listing their endpoints. The true timing path(s) between these endpoints must be identified because the structural STA algorithms may not select the true timing path through the circuit. Some of the paths may have cycles. The RT timing endpoints, along with the true timing paths, are provided as an input to this work. False paths are removed from the timing graphs by cutting timing arcs in the timing graph while preserving the true timing paths to ensure that the STA tools produce correct results. A separate CAD tool performs the timing path preserving cycle cutting operation [4].

Since timing validation can not be performed in a single pass, the RT constraints are grouped into compatibility sets. One set is used for timing driven synthesis and physical design. The full set of RT constraints are employed for timing validation. When performing timing validation, the full path delays are calculated along with margins associated with each RT constraint.

A delay violation exists in a RT constraint when the margin is less than the min-delay minus the max-delay. The design will be signed-off if there are no violations. If timing violations are identified in the validation process, the delay targets associated with the RT constraints need to be updated and rerun synthesis and/or physical design or perform an ECO.

A. Graph Representation of a circuit

Each asynchronous pipeline controller is represented as a cyclic graph $G = (V, E)$ where the input pins of each gate, the primary inputs, and the primary outputs are vertices (nodes) $v_i \in V$ of the graph, and edges $e_i = (v_x, v_y) \in E$ map connectivity between the vertices. The primary input and output vertices of G are identified with a double circle in figures. Fig. 6 is a directed graph representation of the sequential asynchronous handshake pipeline controller in Fig. 5 used to design the circuit in Fig. 4.

A *path* is a sequence of vertices connected by edges in E. The starting and ending nodes of a path are called *timing endpoints*. There can be multiple paths between timing endpoints. One of the paths between the timing endpoints lr and rr is $[lr, 1/A0, 2/A, 5/A0, 3/A2, 4/A, rr]$. A *cycle* is a path that starts and ends with the same vertex. For instance, path $[1/B1, 2/A, 1/B1]$ is a cycle.

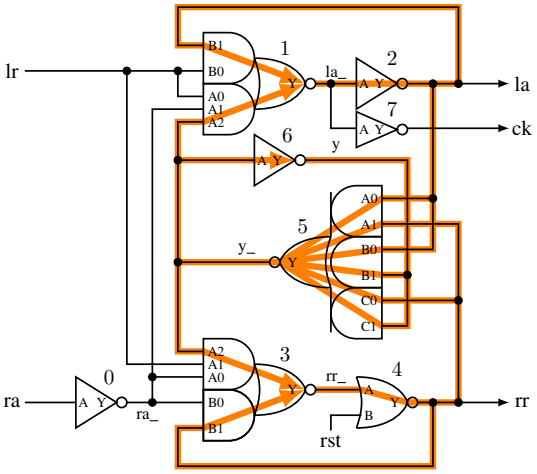

Fig. 5. Circuit implementation of a burst-mode linear controller. The cycles in the design have been highlighted.

B. Identify timing paths for each controller

A bundled data asynchronous design consists of a control network and a datapath as shown in Fig. 2. The control path is formed using pipeline controllers (LC). A set of RT constraints are associated with each controller and these constraints are mapped to the full design.

The RT constraints and associated true paths for the controller of Fig. 5 are shown in Table I. Each relative timing constraint identifies two timing path sets: max-delay path(s) from pod to poc_0 and min-delay path(s) from pod to poc_1. The min-delay path must be at least m time units greater than the max-delay path. The min-delay and max-delay paths for RT constraints (2) and (3) in Table I are both wholly contained within the controller. A constraint may specify multiple true paths, as is the case for the min-delay path of (1) and the max-delay path of (8). The min-delay paths of constraints (0) and (1a) contain cycles because nodes $4/A$ and $2/A$ appear twice in the respective paths. These cycles are caused by the system-level interconnect of the handshake channel and the four-cycle RTZ handshake protocol. RT constraints (8) and

(9) have timing endpoints at the datapath which are external to the controller. These two constraints represent the setup and hold time of the register or latch.

C. True Timing Path Driven Cycle Cutting

Asynchronous controllers contain cycles which must be cut to perform timing driven optimization and static timing analysis. These cycles must be cut in such a manner that the true timing paths in the circuit remain uncut. For example, the cycle $[1/B1, 2/A, 1/B1]$ can be cut at edge $(1/B1, 2/A)$ or $(2/A, 1/B1)$. If the latter cut is employed, all timing paths passing through gate 2 will also be cut. Thus the preferred cut point for this cycle is the edge $(1/B1, 2/A)$.

In this paper, we assume that all the cycles through linear controllers are cut in the design. An external tool is employed to create a DAG when given a Verilog controller along with the associated set of true and false paths. A set of cut points are produced which create acyclic timing graph that cuts all the cycles and false paths, while preserving the true paths. The tool also gives cut paths which will remove system level cycles in the design as described below.

In order to accurately gather the delay value of a cyclic RT constraint, at least two iterations of STA are required. We implement an algorithm to partition the design into independent timing runs that can be composed to create accurate full path timing, including paths with cycles. This is performed by partitioning the constraints into two partitions: forward cycle cut (FCC) to preserve downstream ($\$i1 \rightarrow \$i2$) paths, and backward cycle cut (BCC) to preserve upstream ($\$i1 \rightarrow \$i0$) paths. This works due to the locality of the timing constraints that are tied to each individual controller.

Each linear controller in a design will have an upstream and downstream channel connecting it to the adjacent pipeline stages, as shown in Fig. 2. Consider controller LC_1, and the cycles in the channel connecting it to LC_0 and LC_2. After applying FCC (where timing path $ra \rightarrow rr$ is cut) and BCC (where timing path $lr \rightarrow la$ is cut), we obtain the DAGs shown in Fig. 7 and Fig. 8, respectively.

A search algorithm is implemented to generate overlapping cut points to allow more accurate delay calculations of paths that must be cut to create a DAG. For instance, path

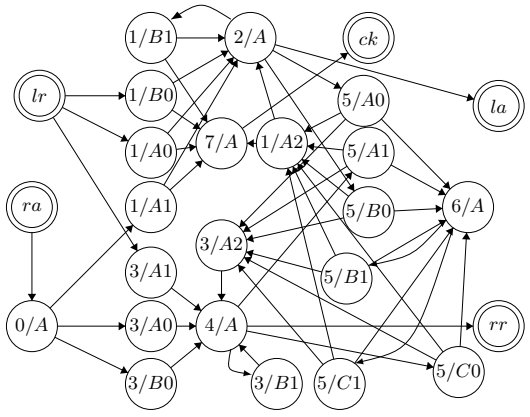

Fig. 6. Graph Representation for the linear controller in Fig. 5

TABLE I
RELATIVE TIMING CONSTRAINTS AND TRUE PATHS FOR FIG. 5 CIRCUIT

RT constraint	Full Timing Path	
rr+ ↦ y+ ≺ rr-	4/A-, 5/A1+, 6/A-, 5/B1+ < 4/A-, rr+, $i2/lr+, $i2/la+, ra+, 0/A+, 3/B0-, 4/A+, 5/A1-	(0)
lr+ ↦ y+ ≺ la-	lr+, 3/A1+, 4/A-, 5/A1+, 6/A-, 5/C1+ < lr+, 1/A0+, 2/A-, la+, $i0/ra+, $i0/rr-, lr-, 1/B0-, 2/A+, 5/A0-	(1a)
	lr+, 1/A0+, 2/A-, 5/A0+, 6/A-, 5/C1+ < lr+, 1/A0+, 2/A-, la+, $i0/ra+, $i0/rr-, lr-, 1/B0-, 2/A+, 5/A0-	(1b)
y_+ ↦ y- ≺ la+	5/C0-, 6/A+, 5/B1- < 5/C0-, 1/A2+, 2/A-, 5/B0+	(2)
y_+ ↦ y- ≺ rr+	5/C0-, 6/A+, 5/C1- < 5/C0-, 3/A2+, 4/A-, 5/C0+	(3)
lr+ ↦ ck+ ≺ la_+	lr+, 1/A0+, 7/A-, ck+ < lr+, 1/A0+, 2/A-, la+, $i0/ra+, $i0/rr-, lr-, 1/B0-, 7/A+	(4)
lr- ↦ ck- ≺ la_-	lr-, 1/A0-, 7/A+, ck- < lr-, 1/A0-, 2/A+, la-, $i0/ra-, $i0/rr+, lr+, 1/B0+, 7/A-	(5)
lr+ ↦ rr+ ≺ lr-	lr+, 3/A1+, 4/A-, 3/B1+ < lr+, 1/A0+, 2/A-, la+, $i0/ra+, $i0/rr-, lr-, 3/A1-	(6)
lr+ ↦ la+ ≺ ra+	lr+, 1/A0+, 2/A-, 1/B1+ < lr+, 3/A1+, 4/A-, rr+, $i2/lr+, $i2/la+, ra+, 0/A+, 1/A1-	(7)
lr+ ↦ $i2R/D ≺ $i2R/CLK-	lr+, 1/A0+, 7/A-, ck+, $i1R/CLK+, $i1R/Q, $i2R/D, < lr+, 3/A1+, 4/A-, rr+, $i2/lr+, $i2/la+, ra+, 0/A+, 3/B0-, 4/A+, rr-, $i2/lr-, $i2/ck-, $i2R/CLK-	(8a)
	lr+, 3/A1+, 4/A-, rr+, $i2/lr+, $i2/ck+, $i2R/CLK+, $i2R/Q < lr+, 3/A1+, 4/A-, rr+, $i2/lr+, $i2/la+, ra+, 0/A+, 3/B0-, 4/A+, rr-, $i2/lr-, $i2/ck-, $i2R/CLK-	(8b)
lr- ↦ $i1R/CLK- ≺ $i1R/D	lr-, 1/A0-, 7/A+, ck-, $i1R/CLK- < lr-, 1/A0-, 2/A+, la-, $i0/ra-, $i0/ck+, $i0R/CLK+, $i0R/Q, $i1R/D	(9)

$[rr+, \$i2/lr+, \$i2/la+, ra+, 0/A+, 3/B0-, 4/A+, rr-]$ is a cycle. The total delay of the path can be computed by cutting it at ra and calculating the timing from $rr+ \rightarrow ra+$, then cutting the path at $\$i2/la$ and calculating delay from $\$i2/la+ \rightarrow rr-$, adding the two delays, then subtracting the $\$i2/la+ \rightarrow ra+$ delay from the total.

D. Graph Coloring Algorithm

An algorithm is written to break all paths with cycles into overlapping acyclic path segments. This generates a set of path segments and timing cut points which are composed to accurately time the full path.

A greedy graph coloring algorithm is applied to group non-intersecting true paths into sets for each controller instances in the design [7]. Two timing paths are intersecting if an

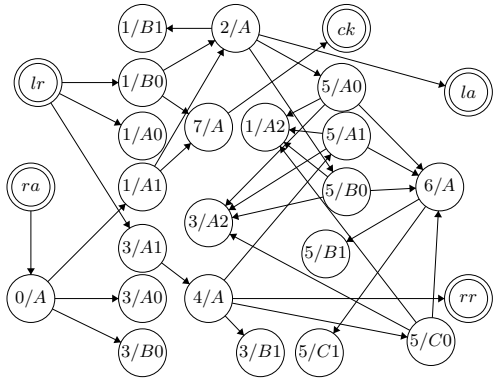

Fig. 7. A DAG of the Fig. 6 timing graph where local cycles have been cut and handshake channel cycles are removed with forward cycle cutting (FCC) by cutting all timing paths between ra and rr.

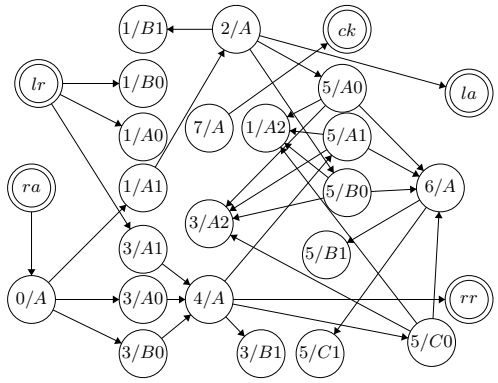

Fig. 8. A DAG created from the timing graph of Fig. 6 where local cycles are cut and BCC is applied to cutting all timing paths between lr and la.

endpoint of a timing path is also an internal node of another timing path. The algorithm ensures that a path in the set will not introduce a cut point in another path in the same set. Partitioning timing paths reduces the number of STA iterations required to generate full-path timing.

Table II reports the results of cutting cycles and partitioning path segments for the RT constraints for the controller shown in Fig. 5. Each set contains a number of non-intersecting paths.

Asynchronous designs can be constructed using more than one type of linear controller. Each controller type will have its associated partition table. The RT constraints are applied to each controller instance in a design. The design shown in Fig. 4 is built using a single type of controller so it requires only one partition table. The RT constraints are applied on each controller instance with independent delay targets.

All the path delays in each set can be analyzed in one iteration. However, when analyzing a pipelined design, the odd controllers (LC_0,LC_2) and even controllers (LC_{10},LC_{11}) of Fig. 4 must be evaluated in separate STA runs. Because the same constraint sets are employed for all the controllers, some external constraints would overlap its adjacent controllers' constraint sets. The number of STA iterations required to perform timing validation would be twice the largest number of constraints with a controller.

The results report the implemented algorithm that cut paths into overlapping segments (including with and without cycles when necessary), partition path segments into compatible sets, and partition controllers into odd and even sets.

E. Perform Static Timing Analysis

Static timing analysis is performed after synthesis or layout. A commercial tool such as Synopsys PrimeTime or Candence Tempus is invoked to perform STA. This paper uses PrimeTime. Multi-mode multi-corner analysis is performed to incorporate process, voltage, and temperate variation on timing path delays. Timing path delays from the complete set of paths (e.g. Table II) are stored into a database.

F. Evaluate RT margin for every timing path

The timing margin for the RT constraint is the amount of time difference between the max-delay and min-delay timing

TABLE II
RELATIVE TIMING GRAPH NODES AS TIMING PATHS

Set	No.	Constraint
A	1	(5/C0-, 6/A+, 5/B1-)
	2	(5/C0-, 6/A+, 5/C1-)
	3	(lr+, 1/A0+, 7/A-, ck+, $i1R/CLK+, $i1R/Q, $i2R/D)
	4	(lr+, 3/A1+, 4/A-, rr+, $i2/lr+, $i2/la+, ra+, 0/A+, 3/B0-)
	5	(lr+, 3/A1+, 4/A-, 3/B1+)
B	1	($i0/ra+, $i0/rr-, lr-, 3/A1-)
	2	(3/A2+, 4/A-, 5/C0+)
	3	(5/C0-, 1/A2+, 2/A-, 5/B0+)
C	1	(lr-, 1/A0-, 7/A+, ck-)
	2	(lr+, 1/A0+, 7/A-, ck+)
	3	(4/A-, 5/A1+, 6/A-, 5/B1+)
	4	(4/A-, rr+, $i2/lr+, $i2/la+, ra+, 0/A+, 3/B0-)
	5	(lr+, 3/A1+, 4/A-, 5/A1+, 6/A-, 5/C1+)
	6	(lr-, 1/A0-, 2/A+, la-, $i0/ra-, $i0/ck+, $i0R/CLK+, $i0R/Q, $i0R/D)
	7	(lr+, 1/A0+, 2/A-, la+, $i0/ra+, $i0/rr-)
D	1	(0/A+, 3/B0-, 4/A+, 5/A1-)
	2	(la-, $i0/ra-, $i0/rr+, lr+, 1/B0+, 7/A-)
	3	(la+, $i0/ra+, $i0/rr-, lr-, 1/B0-, 7/A+)
	4	(la+, $i0/ra+, $i0/rr-, lr-, 1/B0-, 2/A+, 5/A0-)
	5	(0/A+, 3/B0-, 4/A+, rr-, $i2/lr-, $i2/ck-, $i2R/CLK-)
E	1	(lr+, 3/A1+, 4/A-, rr+, $i2/lr+, $i2/la+, ra+, 0/A+, 1/A1-)
	2	(lr+, 1/A0+, 2/A-, la+, $i0/ra+)

paths. These margins are calculated, stored, and compared against the minimum value required for the design. A value less than the required value indicates a failed timing constraint (negative slack). Negative slack must be fixed by changing circuit timing by modifying timing targets, rerunning synthesis or place and route, and then this timing validation tool.

TABLE III
RT SLACK EVALUATION FOR MULTIPLIER DESIGN IN FIG. 4

RT Constraint	pod	poc_0	poc_1	RT slack (ns)
rr+ ↦ y+ ≺ rr-	LC_0/rr	LC_0/y	LC_0/rr	0.97
	$LC_1$0/rr	LC_{10}/y	LC_{10}/rr	1.05
	$LC_1$1/rr	LC_{11}/y	LC_{11}/rr	0.84
lr+ ↦ y+ ≺ la-	LC_{10}/lr	LC_{10}/y	LC_{10}/la	0.97
	LC_{11}/lr	LC_{11}/y	LC_{11}/la	0.9
	LC_2/lr	LC_2/y	LC_2/la	1.07
y_+ ↦ y- ≺ la+	LC_0/y_	LC_0/y	LC_0/la	0.12
	LC_{10}/y_	LC_{10}/y	LC_{10}/la	0.09
	LC_{11}/y_	LC_{11}/y	LC_{11}/la	0.09
	LC_2/y_	LC_2/y	LC_2/la	0.11
y_+ ↦ y- ≺ rr+	LC_0/y_	LC_0/y	LC_0/rr	0.12
	LC_{10}/y_	LC_{10}/y	LC_{10}/rr	0.09
	LC_{11}/y_	LC_{11}/y	LC_{11}/rr	0.09
	LC_2/y_	LC_2/y	LC_2/rr	0.11
lr+ ↦ ck+ ≺ la_+	LC_{10}/lr	LC_{10}/ck	LC_{10}/la_	1.12
	LC_{11}/lr	LC_{11}/ck	LC_{11}/la_	1.07
	LC_2/lr	LC_2/ck	LC_2/la_	1.18
lr- ↦ ck- ≺ la_-	LC_{10}/lr	LC_{10}/ck	LC_{10}/la_	1.12
	LC_{11}/lr	LC_{11}/ck	LC_{11}/la_	1.07
	LC_2/lr	LC_2/ck	LC_2/la_	1.18
lr+ ↦ rr+ ≺ lr-	LC_{10}/lr	LC_{10}/rr	LC_{10}/lr	1.02
	LC_{11}/lr	LC_{11}/rr	LC_{11}/lr	1
	LC_2/lr	LC_2/rr	LC_2/lr	0.9
lr+ ↦ la+ ≺ ra+	LC_0/lr	LC_0/la	LC_0/ra	1
	LC_{10}/lr	LC_{10}/la	LC_{10}/ra	0.91
	LC_{11}/lr	LC_{11}/la	LC_{11}/ra	1.09
lr+ ↦ $i2R/D ≺ $i2R/CLK-	LC_0/lr	R10/D	R10/CLK	0.31
	LC_0/lr	R11/D	R11/CLK	0.31
	LC_{10}/lr	R2/D	R2/CLK	0.45
	LC_{11}/lr	R2/D	R2/CLK	0.25
lr+ ↦ $i2R/CLK- ≺ $i2R/D	LC_0/lr	R10/CLK	R10/D	0.76
	LC_0/lr	R11/CLK	R11/D	0.77
	LC_{10}/lr	R2/CLK	R2/D	0.78
	LC_{11}/lr	R2/CLK	R2/D	0.77

Table III shows the RT slack values obtained for the multiplier design in Fig. 4. The pod column is the timing start point, the poc_0 column contains maximum delay endpoint, and the poc_1 column contains endpoint for the minimum delay path for the specified RT constraint. The slack of each RT constraint is plotted as a histogram in Fig. 12. Constraint

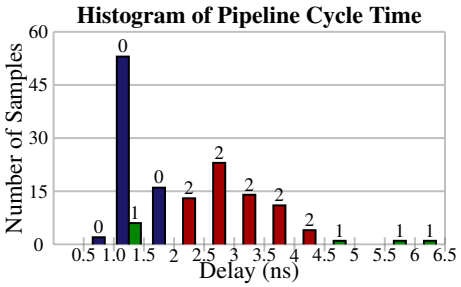

Fig. 9. Cycle time distribution

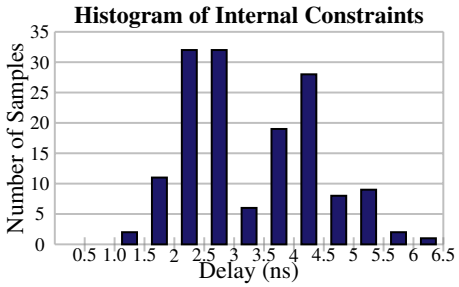

Fig. 10. Distribution of internal Slack

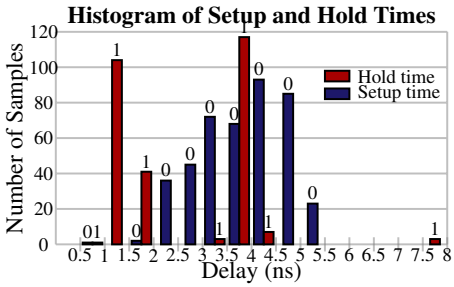

Fig. 11. Slack for External RT constraints

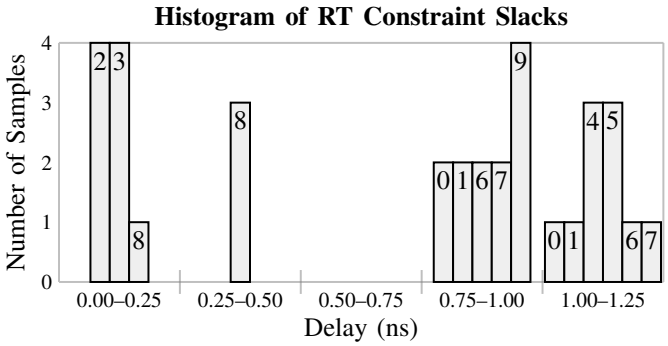

Fig. 12. Slack distribution for Table I constraints applied to Fig. 4 design.

8 is the data setup, 9 is the hold time, and the rest make hazards unreachable. This data allows designers to trade off performance and yield of a design.

IV. Results

The methodology is now demonstrated on a 16-point Fast Fourier Transform (FFT-16) design [8]. The histogram in Fig. 9 shows the cycle time distribution for various categories of pipeline stages in this multi-frequency design. The three different color groups in the graph corresponds to three different operational frequencies in the FFT-16 design. The slack distribution for RT constraints which are required for circuit correctness are shown in Fig. 10. Fig. 11 shows the setup time (blue) and hold time (red) slack distribution for the design. Due to the multi-frequency nature of the design some paths are expected to have large setup and hold times.

TABLE IV
Validation Result

	Multiplier [6]	FFT-16 [8]
Pipe depth	3	16
# of RT constraints	19	1082
Tech Node (nm)	65	180
Area (gates)	1K	57K
Total Run Time(s)	25.8	567.0
Commercial STA (s)	21.8	455.8
Validation (s)	4.0	111.2

Summary information for applying the validation flow on the two designs used in this paper are in Table IV. The complexity of the design is identified based on pipeline depth, number of RT constraints applied, and physical design area. The entire validation run-time is broken into the commercial STA tool run time and our algorithms to create timing set partitions, build full-path timing targets for STA, and collect results.

V. Conclusion

Timing is the primary difference between asynchronous and clocked designs. Traditional static timing analysis algorithms can not be directly applied to asynchronous designs due to cycles and conflicting timing paths. A methodology and algorithm is presented which, when provided timing path information, allows arbitrary timing paths, including those with cycles, to be evaluated using commercial static timing analysis tools such as PrimeTime. Total run times are larger than for a comparable clocked design as multiple iterations through the STA tool are required, but the runs can all be performed concurrently. The paper demonstrates algorithms performing full cyclic path timing verification of a 57K-gate 16 point FFT design in under 10 minutes total run time.

VI. Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 1218012 and Semiconductor Research Corporation under GRC Task 2235.001.

References

[1] R. Nair, L. Berman, P. S. Hauge, and E. J. Yoffa, "Generation of Performance Constraints for Layout," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 8, pp. 860–874, 1989.

[2] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design – A Systems Perspective*. Kluwer Academic Publishers, 2001.

[3] C. J. Myers, *Asynchronous Circuit Design*. J. Wiley, 1999.

[4] K. S. Stevens and V. Vij, "Cycle Cutting with Timing Path Analysis," U.S. Patent No. 8,239,796, Assignee: University of Utah Research Foundation, 29 Jan 2013.

[5] K. S. Stevens, R. Ginosar, and S. Rotem, "Relative Timing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 11, pp. 129–140, Feb. 2003.

[6] K. S. Stevens, Y. Xu, and V. Vij, "Characterization of Asynchronous Templates for Integration into Clocked CAD Flows," in *15th International Symposium on Asynchronous Circuits and Systems*. IEEE, May 2009, pp. 151–161.

[7] J. C. Culberson, "Iterated Greedy Graph Coloring and the Difficulty Landscape," University of Alberta, Dept. of Computing Science, Tech. Rep. TR92-07, 1992.

[8] W. Lee, V. S. Vij, A. R. Thatcher, and K. S. Stevens, "Design of Low Energy, High Performance Synchronous and Asynchronous 64-Point FFT," in *Design, Automation and Test in Europe (DATE)*. IEEE, Mar 2013, pp. 242–247.