Qualifying Relative Timing Constraints for Asynchronous Circuits

Jotham Vaddaboina Manoranjan   Kenneth S. Stevens
Department of Electrical and Computer Engineering
University of Utah

***Abstract*—Relative Timing uses path based timing constraints to guarantee that a circuit conforms to its behavioral specification. Timing constraints are used to order signal transitions or events in a circuit through corresponding minimum and maximum delay timing constraints. A circuit may have multiple sets of constraints, each of which, when satisfied, can individually ensure functional correctness. This paper presents a framework to evaluate and rank relative timing constraint sets for a given circuit. The constraint sets are evaluated on the basis of robustness of the constraints and conflicts between constraints in the same set. The analysis is automated by building a tool. The paper applies the methodology and tool to optimize the extraction of relative timing constraints for delay insensitive timing models of asynchronous circuits. This is demonstrated using a burst-mode controller. The optimization leads to an average tool runtime reduction of 94%.**

I. Introduction

Relative timing (RT) is a formalism that explicitly represents timing constraints necessary for a circuit to operate correctly [1]. This is a unifying method that can represent timing in clocked or asynchronous circuits. RT uses path based timing constraints to make hazards in the circuit unreachable and to ensure that the circuit conforms to its specification. Relative timing based asynchronous design has been shown to provide power, performance and area benefits [2], [3]. Some relative timing based asynchronous designs have shown 3×, 1.5× and 1.2× benefits in terms of energy, area and performance respectively when compared to synchronous versions.

Often there are multiple relative timing constraints associated with a circuit. Together these constraints form a *set* of relative timing constraints. Every individual constraint within a constraint set must be jointly enforced. Since these constraints are timing constraints that fundamentally effect circuit delays, they directly impact circuit performance. Furthermore, the margin within which the constraints are satisfied directly affects the robustness of the circuit.

When multiple constraint sets can be implemented, each of which guarantee functional correctness of the circuit, a choice needs be made between the constraint sets. At present no methodology exists to evaluate and choose between multiple constraint sets. This paper defines an approach to evaluate and rank RT constraint sets for a circuit. There are two primary considerations for evaluation of such constraint sets. First, the robustness of the constraints based on the paths delays must be considered. Second, the difficulty of implementing the constraints, such as when two-sided constraints (both a maximum delay and a minimum delay constraint) are identified [4]–[6].

The work presented here creates a methodology to evaluate constraint sets based on robustness and difficulty of implementation. An evaluative analysis is used to assess the quality and potential implementability of RT constraint sets. The analysis is then incorporated into a tool to optimize the generation of RT constraints for delay-insensitive (DI) circuit models. Beginning with a simpler speed-independent (SI) circuit model, wire forks are iteratively added to the circuit to create a delay-insensitive model. After each iteration the best constraint set, as picked by the proposed analysis is used as the input for the next iteration. This leads to a reduction in overall tool runtime for the generation of RT constraints.

II. Background

The relative timing methodology has been successfully applied to create functionally correct bundled data based asynchronous systems and also maps well to FPGA designs which have large wire delays due to the FPGA routing structures [7], [8]. The formal representation of a simple RT constraint is defined in Eqn. 1. It specifies that the maximum delay (max-delay) from a common point-of-divergence pod to the point-of-convergence poc_0 is less than the minimum delay (min-delay) between the pod and poc_1. Thus relative timing constraints orders events, causing poc_0 to always precede poc_1.

$$pod \mapsto poc_0 \prec poc_1 \tag{1}$$

Relative timing provides a methodology to model, verify, and implement circuits and protocols that are not delay-insensitive or speed-independent. A timed circuit can be proven conformant to a formal specification of the design when a set of RT constraints are applied to the design. If these constraints are faithfully implemented, hazards and failures due to timing will not occur.

There are numerous sets of relative timing constraint sets that can be applied to a design to make it conformant to a specification. Different locations can be used for many pod and poc timing endpoints, with differing timing margins and relationships to the other constraints in a complete RT set. Constraint sets can be automatically generated using the the ARTIST tool [9]. Given a circuit implementation and a formal design specification, ARTIST automatically generates sets of relative timing constraints, that when applied to the implementation, proves conformance to the specification. Internally this tool performs a breadth first or depth first search to generate sets of timing constraints. Constraints are heuristically selected based on a set of internal rules. A large set of constraints can be found based on the search algorithm and heuristics employed. However, the set of constraints will not create the full set of possible RT constraint sets that make a design conformant to the specification. The goal is to find a relative timing constraint set that can create a robust circuit implementation.

Bundled data design is partitioned into a control path and a data path. The data path of the system consists of combinational logic and registers, similar to that in a synchronous pipeline. The control path replaces the global clock and is composed of modules that carry out request-acknowledge (req-ack) handshaking, controls frequency of operation, and clocks the latches. The control logic maintains the timing and functional relationship between pipeline stages. It is imperative that these handshake signals are hazard free because a glitch in a handshake signal can be interpreted as a valid signal transition causing a miscommunication between controllers. This can result in the circuit settling in an unwanted state which can be fatal to circuit operation. This paper applies relative timing to bundled data based asynchronous systems which have timing constraints inside the controllers and between the controllers and the data path.

III. Motivation

Constraint sets rarely consist of a single relative timing constraint (RTC). The size of these sets can vary depending on circuit complexity; however even simple circuits can produce constraint sets with 5–10 constraints. A choice has to be made between the presented sets of RTCs. Simple rudimentary techniques can be applied to select a constraint set, such as the total number of constraints in a set or the number of gates in each path. However such techniques fail to take into account any information provided by either the circuit structure or the constraints themselves. It is easy to see how a constraint set with numerous "easy to implement constraints" would be preferable over a set with a few "hard to implement constraints".

Hence a qualifying methodology that identifies and ranks constraint sets is tremendously beneficial. Being able to identify and discard bad constraint sets early in the design process can help save significant design time. Furthermore, a defined quality metric will lead to higher degrees of design automation. Qualitative analysis of constraints can also enable better circuit design choices. For instance, suppose a choice needs to be made between two possible asynchronous controllers with associated RTCs, the choice can be driven by an analysis of the constraints associated with each of the controllers, identifying the more robust and efficient controller.

The work presented in this paper defines and automates a quality metric.

IV. Identifying the Issues

There are two primary circuit paradigms that have been taken into account in the paper. These are introduced briefly in this section.

A. Robustness

As discussed earlier RTCs order signals by enforcing certain timing requirements on the circuit. For Eqn. 1 to be satisfied the path delay from pod to poc_0 (referred to as the early path) must be less than the path delay from pod to poc_1 (referred to as the late path). The margin, or difference in delay between the early and late paths, is a measurement of robustness. Intuitively it can be concluded that it is preferable to have a longer late path with more gates (and possibly wire delay) and a shorter early path. This will enable easy satisfaction of the constraint in the circuit implementation and greater robustness. However suppose the converse is true of the constraint, i.e., the late path has fewer gates than the early path. Satisfying the constraint may require the addition of delay elements in the late path. This can effect overall circuit performance, power and area. Hence, given a choice, the former may be preferred over the latter.

B. Timing Conflicts

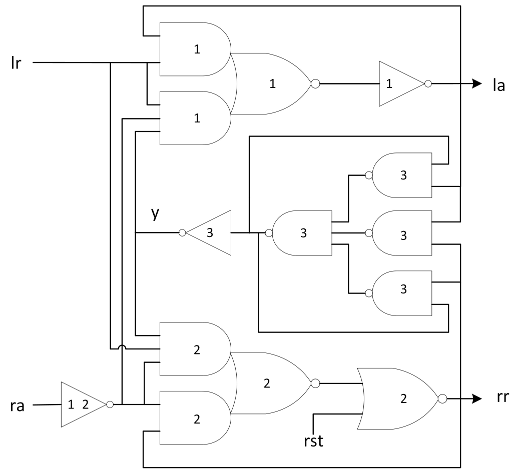
Each constraint set may have numerous constraints, and individual RTCs in a set may traverse common paths. This makes it likely that the implementation of one constraint may affect implementation of another constraint. Such common paths may create a degree of interference or conflict between constraints, that in turn may limit the margin within which the two constraints can be satisfied. Some constraints may be two-sided [4]. In extreme cases, the implementation of one timing constraint may make it impossible to implement another timing constraint.

Constraint conflicts can significantly impact the implementability of a constraint set. For instance two constraints may place competing timing requirements on a single path. Even though there may be a margin within which both constraints could be satisfied, a penalty is borne in design effort and time. The designer would benefit from using an alternative constraint set that does not contain conflicting constraints, enabling more robust and efficient designs.

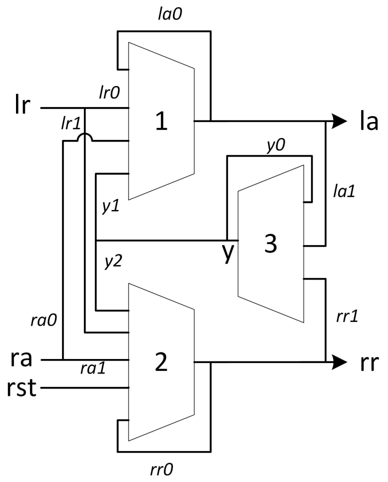
V. Circuit and Scope

A. Circuit

While the work being presented in this paper is relevant to circuit implementation on both ASICs and FPGAs, the impact on asynchronous circuit design on FPGAs is possibly more profound. This is primarily due to the fact that, given the logic and routing structures, it is more challenging to enforce timing requirements on FPGAs. In an ASIC, the wire delay inside a single controller from the output of a gate with fanout to multiple gate inputs is usually in an equipotential region because local wire delays are small relative to gate delays. Here the speed-independent timing model is applicable. Unlike ASICs, signals that drive multiple functions typically pass through routing structures that result in vastly different delays. Signal delays on FPGAs are significantly higher than on ASICs and are comparable to logic delays. Therefore, the timing model for FPGAs typically requires a delay-insensitive model. Hence signal path analysis for robustness will have greater impact on



(a) ASIC logic mapping of a burst-mode controller



(b) FPGA mapping with labeled wire forks

Fig. 1: Burst-mode controller implementation

FPGAs. Since FPGA design is more challenging, we apply our approach to FPGA implementations.

The burst-mode controller illustrated in Fig. 1a is used to describe the proposed analysis. The boolean logic of the controller in Fig. 1a translates to the circuit structure depicted in Fig. 1b on an FPGA. There are three LUTs shown, each LUT absorbs the logic elements as labeled in the figures. The implementation is then verified against the behavioral specification of the circuit and sets of RT constraints are generated [9]. Signal forks on FPGAs are rarely isochronic and signals can assume any order of arrival at the destinations. Hence, in Fig. 1b, it is not possible to guarantee that the delay associated with *y1* is the same as the delay on *y2*. Hence they are labeled independently to indicate differing delays.

B. Scope

The analysis presented here is intended to be applied to the early part of the design process where an RTC set is being chosen for a circuit implementation. This work does not address the final circuit level implementation of RT constraints which depend on other factors beyond the those discussed here. Factors such as variations on a chip due to process, temperature and voltage also impact circuit robustness. However, these factors are excluded from the analysis as the scope of the paper is to qualify RTC sets relative to each other. The proposed metric is best applied to situations where multiple RT constraint sets are being evaluated against each other for a given circuit implementation structure.

VI. Analyzing Robustness of Constraint Sets

The first step in the analysis is to identify the "inherent robustness" of the constraint sets. This is measured by the difference in the lengths of the late path and early path of an RT constraint. For instance, the following is a constraint generated for the implementation of the circuit in Fig. 1b.

$$lr0 \mapsto la0 \prec y1 \tag{2}$$

Eqn. 2 orders the signal transitions such that $la0$ occurs before $y1$, which both follow a transition on $lr0$. This RTC is converted into the mathematical inequality of Eqn. 3 for an FPGA, which when met satisfies the constraint. The delay associated with each signal wire is represented by its label in Fig. 1b. The delays associated with the LUTs labeled 1 and 3 are represented as $d1$ and $d3$ respectively. A margin of m is now additionally incorporated.

$$lr0 + d1 + la0 + m \leq lr0 + d1 + la1 + d3 + y1 \tag{3}$$

The inherent robustness is measured as the difference in the number of logic delays and signal delays between the RHS and the LHS of the above inequality. A coarse approximation allows a single nominal delay value D_L to be assigned to each LUT and D_S to each signal fork. Thus D_S models $lr0$, $la0$, $la1$ and $y1$, and D_L models $d1$ and $d3$. Robustness is directly linked to the margin. The inherent robustness is represented by the value of m obtained. For the above equation,

$$\begin{aligned}
&m = (lr0 + d1 + la1 + d3 + y1) - (lr0 + d1 + la0)\\
&m = (D_S + D_L + D_S + D_L + D_S)\\
&\quad -(D_S + D_L + D_S)\\
&m = D_S + D_L
\end{aligned}$$

The higher the value to m the better the robustness of the constraint. Each constraint in the set is analyzed and the value of m is calculated. Both the average and the worst-case value of m in a set are used to qualify the constraint set.

In the circuit being discussed, there is a causal relationship between output la and input lr, and output rr and input ra. An event on lr is triggered by the environment in reaction to an event on la. This is part of the handshaking protocol of the controller. In certain cases the RTC's signal paths traverse these output-input pairs. In these cases, the environment is assumed to have a single logic delay D_L associate with it.

VII. Analyzing Constraint Timing Conflicts

A. Defining Interactions and Conflicts

RTCs can be used to optimize an implementation. Synthesis and place and route tools may need to reduce the max-delay of the early path $pod \mapsto poc_0$ and increase the min-delay of the late path $pod \mapsto poc_1$ for a constraint to have sufficient robustness.

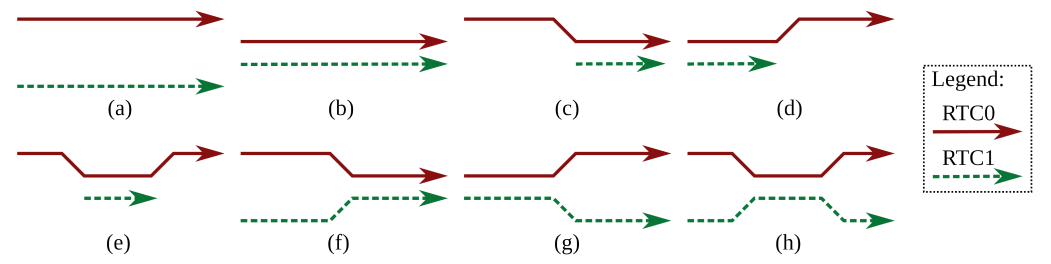


Fig. 2: Examples of various ways in which paths from different RTCs can overlap.

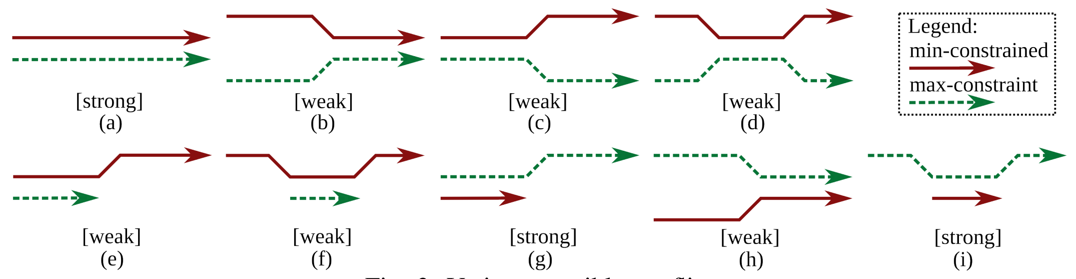


Fig. 3: Various possible conflicts

Consider the following RTCs: $rtcA : x \mapsto y \prec z$ and $rtcB : x \mapsto w \prec y$. In this case $rtcA$ may be attempting to reduce delay in the path $x{\mapsto}y$ while $rtcB$ is trying to increase the path delay. This creates a conflict.

Depending on the circuit structure, timing targets, and other constraints in a set, it may be possible that conflicting constraints could be simultaneously satisfied. However, there may be penalties associated in terms of the need for additional delays that may impact overall circuit performance. Hence, the impact of such an interference between the constraints and possible conflicts cannot be ignored and provides valuable insight into the ability to optimize a circuit for the given constraint set.

B. Path Interactions

The variety of timing path interactions are identified by the various ways in which two paths can overlap with each other. Fig. 2a shows two RTCs that do not have any common path segments. Fig. 2b shows two paths that are completely overlapping. They have the same start (pod) and end points ($poc_{0/1}$). Fig. 2c–2e show one path contained in another. Fig. 2f–2h show two paths that have only subsections as common paths.

Fig. 2 shows only elementary paths. RTCs can traverse paths that contain cycles where certain wires and logic elements are exercised multiple times between the pod and poc owing to feedback structures [10], [11]. Therefore the various path interactions are expanded to include paths that contain cycles. Hence a path is defined to be equal to another path based on the gates and signals (or wires) traversed independently from the number of times the path is crossed. Thus if one constraint covers path $[A, X, Y, X, Y, Z]$ and another $[B, X, Y, Z]$ we have the condition covered in Fig. 2f. Thus the paths are evaluated based on the *set* of logic elements and wires contained in the path rather than the sequence exact paths traversed.

Using the set notation, a path can be a sub-path of another, similar to the way in RTC1 is a sub-path of RTC0 in Fig. 2c–2e, if the set of logic elements and wires contained by it is a subset of the logic elements and wires traversed by another path. Further, two paths can interact in ways similar to Fig. 2f–2h if they both traverse common logic elements while the common set does not fully cover the entire set of either paths. The shown interactions aren't exhaustive, but are sufficient for a clear description of the approach. Since the paths are viewed as sets, the sets can either be *equivalent*, *disjoint* or *overlapping*. When they overlap, one can be the subset of another, or they can have common elements in a way that the intersection is not equivalent to either set.

C. Analyzing Conflicts

In each of the possible cases in Fig. 2, a conflict between RTC0 and RTC1 is only created when a late path in one constraint overlaps with an early path in another constraint. If both the RTCs are constraining the paths in the same fashion (either only early or late), no conflict is produced. In the case of only an early path constraint, the smaller max-delay requirement is employed for both constraints, which inherently satisfies the larger value. For instance, consider Fig. 2b. Suppose RTC0 requires the max-delay to be 800 ps while RTC1 requires a delay of 500 ps. RTC0 will be satisfied with increased margin if the the max-delay requirement of 500 ps of RTC1 is employed. Similarly, for two overlapping late path constraints, only the larger value is considered for the min-delay requirement. Hence we exclude these two cases from any further analysis.

In the case where the RTC0 and RTC1 place opposing constraints on interacting paths a conflict is created. This conflict can either be a strong conflict or a weak conflict. A strong conflict occurs when the implementation of one constraint completely covers another. For instance, in Fig. 2b, RTC0 places a max-delay constraint on one path and RTC1 places a min-delay constraint on the exact same path. This would lead to a strong conflict where the implementation of one of the RTCs gives little or no latitude for the implementation of

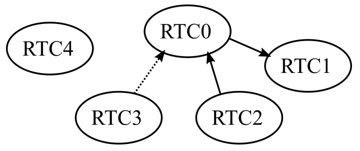


Fig. 4: Graphical representation of constraints and dependencies

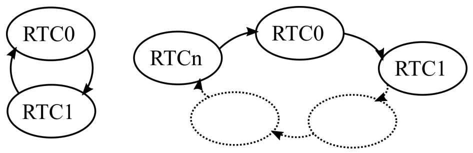


Fig. 5: Graphical representation of cyclical dependencies

the second RTC. While there may be possible delay values which allow both constraints to be satisfied, such conflict can pose major optimization challenges. Another strong conflict can occur, when in Fig. 2c, RTC1 places a min-delay constraint while RTC0 places a max-delay constraint. If the value of the min-delay is greater than that of the max-delay a strong conflict is created as it would not be possible to satisfy RTC0 if RTC1 is satisfied.

On the other hand, consider again Fig. 2c but with RTC0 placing a min-delay constraint, while RTC1 places a max-delay constraint. This would also create a conflict, but what can be perceived as a weak conflict. Even if RTC1 was implemented without any regard for RTC0 and the overlapping path was made to be as fast as possible, there would still be untouched signal paths on RTC0 to which necessary delays could be added to satisfy the min-delay requirements for RTC0. Hence even though implementation of one of the constraints affects or limits the scope of the implementation of another, it does not impair it in totality.

Fig. 3 now proceeds to identify all possible strong and weak dependencies that exist between the paths as shown in Fig. 2 based on early (max-delay) and late (min-delay) path constraints. Fig. 3a, Fig. 3h and Fig. 3j represent strong conflicts, while the rest represent weak conflicts.

This methodology identifies path based timing conflicts based on the difficulty of resolution. The resolution of these conflicts can impact signal or logic delays of a circuit. The addition of delays to a circuit to resolve conflicts will inevitably impact power and possibly performance. Hence this methodology identifies the presence of conflicts within a constraint set in order to pick better constraint sets and possibly avoid the need to add additional delays to a circuit.

D. *Representing Conflicts*

Relative timing constraint conflicts are represented as a directed graph. A directed graph $G = (V, E)$ where V is the set of vertices of G and E is the set of ordered ordered pairs (v_i, v_j) where $v_i \wedge v_j \in V$. For a given constraint set, each individual RTC is represented by a vertex. A conflict between two RTCs is represented by a directed edge connecting the two. The direction of the edge maps from the RTC applying the late min-delay constraint to the RTC with the early max-delay constraint. Furthermore, each edge is further given an attribute that would define the conflict it represents as either a strong conflict or a weak conflict. A strong conflict is represented by a solid directional edge, whereas a weak conflict is represented by a dashed directional edge in the graph.

Fig. 4 shows a directed graph representing a constraint set. The constraint set comprises five constraints represented as vertices (RTC0-4). There exists a strong conflict between RTC0 and RTC1 and between RTC2 and RTC0. There also exists a weak conflict between RTC3 and RTC1.

E. *Cyclical Conflicts*

A critical constraint conflict that can impair circuit implementation is a cyclical conflict. It is possible for two RTCs to have opposite timing requirements on exactly the same set of paths. In this case the early path in one constraint is the late path in another, and vice versa. Thus the realization of one of the constraints makes it impossible to conform to the other constraint. This cyclical conflict can also occur when two or more constraints form a cyclical chain of conflicts.

The graphical representation presents a framework to identify cyclical conflicts between relative timing constraints. A cyclical conflict between RTC constraints is presented when given the RTC directed graph, there exists a path that begins and terminates at the same vertex, traversing at least one other vertex. Fig. 5 shows the graphical nature of such conflicts. The first graph shows a cyclical conflict between two RTC constraints. The second graph shows a cyclical conflict between more than two constraints (RTC0, RTC1 up to RTCn).

Various combinations of strong and weak conflicts can be present in a cyclical path. If a cyclical path exists with all strong conflicts, it is a more significant problem for circuit implementation than for one with only some or no strong conflicts. We shall refer to the former as strong cyclical conflicts, and the latter as weak cyclical conflicts.

VIII. Combining the Analysis

Thus far the paper has presented various metrics that can be extracted from a RTC set associated with a circuit. The presented metrics are:

1) Inherent robustness (m)
2) Weak conflicts (wc)
3) Strong conflicts (sc)
4) Weak cyclical conflicts (wcc)
5) Strong cyclical conflicts (scc)

Most controllers are relatively small circuits. This makes it very probable that every RTC has at least one signal or logic element overlap with another RTC. This leads to a copious number of weak conflicts. We have observed that weak conflicts are not a good indicator of the quality of an RTC set. Therefore we do not use weak conflicts for ranking constraints. Likewise, asynchronous pipeline controllers use combinational feedback to create state. This creates cycles in the RTC graph when at least one relative timing constraint arc passes through each gate. Thus, by the nature of these designs and the large number of weak conflicts, many weak cyclical conflicts are formed. Weak cyclical conflicts may consist entirely of weak conflicts, but may also contain one or more strong conflict. We have found that weak cyclical conflicts that consist entirely of weak conflicts are also not a good indicator of RTC set quality and are thus not considered for ranking constraints. Weak cyclical conflicts with at least one strong conflict segment have been shown to impact the resolution of the strong conflict(s), so these are important for analysis.

Cycles in the RTC graph can intersect and overlap. In this algorithm we minimize the number of cycles that are considered in the RTC graph by assuming that shared conflict edges can resolve multiple cycles. Consider two cyclical conflicts [RTC0, RTC1, RTC2, RTC0] and [RTC1, RTC2, RTC3, RTC1]. They contain an overlapping edge [RTC1, RTC2]. Resolving this overlapping edge will resolve both cycles.

Of the five presented metrics, our algorithm uses inherent robustness m, strong conflicts sc, a subset of weak cyclical conflicts wcc, and strong cyclical conflicts scc.

Each constraint set for a circuit is analyzed and the calculated metrics are used to rank the constraint sets. The designer can use varying priorities between the calculated metrics to choose the best constraint sets. The order of priority used in this paper is presented in the following section as part of the automation.

IX. RT Generation Run Time Improvements

This work automates part of the relative timing characterization process for asynchronous controllers. ARTIST allows us to generate RT constraint sets, but manual intervention has been needed to choose between constraint sets. Run times to generate a complete set of constraints can also be large. ARTIST relies on state space exploration to generate needed constraint sets. The delay-insensitive model of a circuit that is needed for FPGA based implementation creates much larger state spaces, leading to higher RT generation runtime. For example, RT constraint generation runtime for the DI model of the circuit in Fig. 1b took 1.98 CPU hours.

An optimization that reduces this tool runtime is implemented by incrementally adding complexity to the formal circuit model. The circuit is first modeled as speed-independent. ARTIST is then used to verify the circuit and generate constraint sets. The smaller SI model allows for shorter tool runtime. The constraint sets are evaluated based on this methodology and the best constraint set is picked. A single DI signal fork is now added to the implementation model, and the constraint set generated from the SI model is provided as an input to the run. This RT constraint set restricts the search space of the algorithm, and only new timing constraints induced by the DI fork are generated. The resulting RT constraints are evaluated and retained as inputs for the next iterative run and an additional fork is added. This is iteratively done until all required forks are modeled to create a delay-insensitive model of the design.

X. Results: Implementation and Automation

A. Circuit Implementation

The circuit presented in Fig. 1b was verified and ARTIST generated 10 constraint sets. Each of the constraint sets were analyzed and the various metrics were extracted. The first (set0) and third (set2) constraint sets are shown in Fig. 6 and 7. The figures also show the associated inequalities, which are based on the signal traces provided by ARTIST in addition to the RT constraints. Both these constraint sets when implemented for the circuit in Fig. 1b will guarantee the functional correctness of the circuit. The constraint sets have some constraints in common as indicated.

Constraint set2 in Fig. 7 illustrates the importance of why evaluative metric is needed. The constraint $rtc4$ in set2 has a much longer early path (more logic and signal delays) compared to its late path. While this is still implementable, it would have to be at the cost of adding delays on to certain elements on the late path to satisfy the constraint. This adversely impacts circuit power and performance. Analysis of the inherent robustness of the constraint, as proposed in this paper, can identify such poor constraints. Also, constraint set2 illustrates a strong cyclical conflict between $rtc7$ and $rtc8$ which identify constraints with opposing late and early paths.

Delays through the environment are represented by D_E in Fig. 6 and Fig. 7. As pointed out earlier, D_E is assumed to be equal to D_L when calculating the robustness margin m. All constraint sets are analyzed and m values, along with the number of strong conflicts and cyclical conflicts, are calculated. The calculated m values for set0 are presented in Table I. No strong conflicts exist between RTCs. Hence, there are no relevant strong or weak cyclical constraints.

For the circuit described in Fig. 1b, which is an FPGA based realization of the controller, is placed within a SliceL of the Xilinx vc707 FPGA chip (Device: XC7VX485T Package: FFG1761 Speed:-2). All possible placements solutions were exercised, and for each placement the Xilinx FPGA Editor was used to extract signal delays. The delays are then used to check whether or not each constraint was satisfied. When a constraint was satisfied the margin with which it was satisfied was noted. When this margin was negative, the constraint is considered not satisfied. The worst margin among the constraints and the average margin of a constraint set allow us to rank the various placement solutions. The higher the worst case margin, the better the placement solution. Using this, the best placement solution for each of the constraint set was noted. For certain constraint sets, no placement solutions may exist that satisfy all of its constraints.

Table II presents the placement results for the 10 sets of generated constraints. Only the best placement solution with the largest worst margin for the constraints is noted. For each constraint set, the worst-case m value and the average m value are calculated. Table II lists the number of strong conflicts, strong cyclical conflicts and weak cyclical conflicts for each constraint set. The last few columns of the table give the best, worst, and average constraint margins of

$rtc0 : lr \mapsto y0 \prec la1;$
$(lr1 + d2 + rr1 + d3 + y0 + m \le lr0 + d1 + la + D_E + lr0 + d1 + la1)$
$rtc1 : lr1 \mapsto y0 \prec rr1;$
$(lr1 + d2 + rr1 + d3 + y0 + m \le lr1 + d2 + rr + D_E + ra + d3 + rr1)$
$rtc2 : lr1 \mapsto y2 \prec y;$
$(lr1 + d2 + rr1 + d3 + y2 + m \le lr1 + d2 + rr + D_E + ra + d2 + rr1 + d2)$
$rtc3 : lr \mapsto y2 \prec lr1;$
$(lr1 + d2 + rr1 + d3 + y2 + m \le lr0 + d1 + la + D_E + lr0 + d1 + la + D_E + lr1)$

Fig. 6: Relative timing constraint set: set0

$rtc0$ - same as $rtc0$ from Set0
$rtc1$ - same as $rtc1$ from Set0
$rtc2$ - same as $rtc2$ from Set0
$rtc3 : lr \mapsto y1 \prec lr1;$
$(lr1 + d2 + rr1 + d3 + y1 + m \le lr0 + d1 + la + D_E + lr0 + d1 + la + D_E + lr1)$
$rtc4 : lr \mapsto lr1 \prec y2;$
$(lr1 + d1 + la + D_E + lr1 + d1 + la + D_E + lr1 + m \le lr1 + d2 + rr1 + d3 + y2)$
$rtc5 : lr \mapsto y0 \prec rr1;$
$(lr1 + d2 + rr + D_E + ra1 + d2 + rr1 + d3 + y0 + m \le lr0 + d1 + la + D_E + lr0 + d1 + la + D_E + lr1 + d2 + rr1)$
$rtc6 : rr1 \mapsto y2 \prec y;$
$(rr1 + d3 + y2 + m \le rr1 + d3 + y1 + d1 + la1 + d3)$
$rtc7 : lr \mapsto ra1 \prec lr1;$
$(lr0 + d1 + la + D_E + lr0 + d1 + D_E + lr1 + d2 + rr + D_E + ra1 + m \le$
$lr1 + d2 + rr + D_E + ra1 + D_E + rr1 + d3 + y1 + d1 + la + D_E + lr1)$
$rtc8 : lr \mapsto lr1 \prec ra1;$
$(lr1 + d2 + rr + D_E + ra1 + D_E + rr1 + d3 + y1 + d1 + la + D_E + lr1 + m \le$
$lr0 + d1 + la + D_E + lr0 + d1 + D_E + lr1 + d2 + rr + D_E + ra1)$

Fig. 7: Relative timing constraint set: set2

TABLE I: RTC set0 analysis

Constraint#	m
rtc0	$1 * D_S + 1 * D_L$
rtc1	$1 * D_S + 1 * D_L$
rtc2	$2 * D_S + 1 * D_L$
rtc3	$2 * D_S + 2 * D_L$
Worst-case m	$1 * D_S + 1 * D_L$
Average m	$1.5 * D_S + 1.3 * D_L$

the best observed implementation results on the FPGA. The last column indicates whether or not a placement solution satisfying all constraints for each set was found within the slice.

The results indicate that sets 2 through 9 with lower m values and higher degrees of conflicts are never satisfied. No placement solution within the slice is found that satisfies all the constraints. On the other hand, set0 and set1, that have no strong conflicts and have a positive worst-case m, find solutions with robust margins. Furthermore, the strong cyclical conflict in set2 is successfully identified.

B. Automation of Constraint Set Analysis

A tool incorporating the proposed methodology was built to analyze generated RT constraint sets. The metrics were used in the following order of precedence or priority to rank the constraint sets:

1) *Strong cyclical conflicts*: Sets with the lower number of strong cyclical conflicts are ranked higher
2) *Worst case m*: Sets with higher worst case m are ranked higher
3) *Strong conflicts*: Sets with lower number of strong conflicts are ranked higher
4) *Weak cyclical conflicts*: Sets with lower number of weak cyclical conflicts are ranked higher
5) *Average m*: Sets with a higher average m are ranked higher. Usually, this metric is only used to break ties between sets.

The tool takes the circuit structure and relative timing constraints as inputs. The algorithm traverses through each of the constrained paths and calculates the m values. The path analysis is also utilized to construct the RTC graph, which is used for conflict analysis.

C. Optimized RT constraint Generation

The generation of delay-insensitive relative timing constraints described in Sec. IX was automated using the analysis tool to prioritize constraint set selection. The run time of automatic constraint generation and selection of the best constraint set were evaluated. Table III show the runtime results for the circuit shown in Fig. 1b. The run time to generate the constraint sets from a fully delay-insensitive model is reported on the first row. The second row shows the run time for RT constraint set generation, and the analysis tool run time using the iterative constraint generation starting with a

TABLE II: Controller implementation results on the FPGA

RTC Set	Worst-case m	Average m	Strong Conflicts	Strong Cycl. Conflicts	Weak Cycl. Conflicts	Maximum margin (ps)	Minimum margin (ps)	Average margin (ps)	Constraints met?
set0	$1 * D_S + 1 * D_L$	$1.5 * D_S + 1.3 * D_L$	0	0	0	661	212	410	Yes
set1	$1 * D_S + 1 * D_L$	$1.5 * D_S + 1.3 * D_L$	0	0	0	438	317	358	Yes
set2	$-3 * D_S + -3 * D_L$	$0.7 * D_S + 0.4 * D_L$	11	1	9	334	-248	148	No
set3	$-3 * D_S + -3 * D_L$	$0.9 * D_S + 0.7 * D_L$	9	0	9	474	-173	116	No
set4	$-3 * D_S + -3 * D_L$	$0.9 * D_S + 1 * D_L$	7	0	7	532	-173	196	No
set5	$-3 * D_S + -3 * D_L$	$0.9 * D_S + 0.9 * D_L$	7	0	6	532	-173	179	No
set6	$-3 * D_S + -3 * D_L$	$1.1 * D_S + 1 * D_L$	7	0	7	535	-173	267	No
set7	$-3 * D_S + -3 * D_L$	$1.2 * D_S + 1 * D_L$	7	0	7	769	-173	222	No
set8	$-3 * D_S + -3 * D_L$	$1.1 * D_S + 1 * D_L$	7	0	6	769	-173	267	No
set9	$-3 * D_S + -3 * D_L$	$1.3 * D_S + 1.1 * D_L$	7	0	7	665	-173	208	No

TABLE III: RT generation runtime reduction using developed tool for circuit shown in Fig. 1b.

	RT generation runtime (s)	Analysis tool runtime (s)	Total runtime (s)
Direct DI model	7139.52	–	7139.52
Iterative model	8.01	5.85	13.86
Total runtime reduction	–	–	99.8%

TABLE IV: RT generation runtime reduction for asynchronous controllers using developed tool

Controller	Reference	Traditional runtime (s)	Iterative tool runtime (s)	Reduction
LC_BM	[7]	7,139.52	13.86	99.8%
PCHB	[12]	3.56	0.53	85.1%
BRF1,LH1	[13]	10,342.71	297.62	97.2%

speed-independent circuit model. A 99.8% reduction in RT generation runtime time is achieved.

Table IV show a runtime reduction for other controllers that have been discussed in literature. The table also shows the results obtained from the controller implemented in this paper. While RT generation time can significantly vary with factors such as circuit size, state-space and concurrency, the table shows a steady runtime reduction between the traditional approach and the iterative approach. The average reduction in tool runtime was 94%.

XI. Conclusion

A methodology to qualify relative timing constraint sets is presented. The methodology evaluates and ranks numerous possible RT constraint sets for a circuit based on the potential robustness of constraints and the presence of conflicting constraints.

The inherent robustness of a constraint set is calculated in terms of the difference between its late path and early path in terms of signal and logic delays. Various conflicts that could exist between constraints with over-lapping signal paths are identified and categorized based on the impact they may have on circuit implementation. Further, a graph based analytical method is used to identify various cyclical conflicts within constraint sets. The methodology is demonstrated using the implementation of a standard benchmarked burst-mode controller.

The approach was automated and applied to several circuit examples. The approach was validated by implementing all of the constraint sets in an FPGA slice with exhaustive path mappings to find the best solution. Though the methodology is validated on an FPGA fabric, the analysis itself is not dependent on any prior information about the implementation fabric. All that is needed is the circuit structure and the RTC sets. The analysis can be performed for both ASIC and FPGA circuit implementations.

The methodology is used to automate the ranking of constraint sets as part of a custom tool. The tool in incorporated into the RT generation tool flow with incremental addition of signal forks. An average reduction of 94% in RT generation tool runtime is noted with the proposed optimization.

Acknowledgments

This material is based upon the work supported by the National Science Foundation under Grant Number 1111533.

References

[1] 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.

[2] K. S. Stevens, S. Rotem, R. Ginosar, P. Beerel, C. J. Myers, K. Y. Yun, R. Kol, C. Dike, and M. Roncken, “An Asynchronous Instruction Length Decoder,” *IEEE Journal of Solid State Circuits*, vol. 36, no. 2, pp. 217–228, Feb. 2001.

[3] 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.

[4] P. A. Beerel, R. O. Ozdag, and M. Ferretti, *A Designer’s Guide to Asynchronous VLSI.* Cambridge University Press, 2010, ISBN 0521872448,9780521872447.

[5] H. Kim, P. A. Beerel, and K. S. Stevens, “Relative Timing Based Verification of Timed Circuits and Systems,” in *8th International Symposium on Asynchronous Circuits and Systems*. IEEE Press, Apr. 2002, pp. 115–126.

[6] W. Chuang, S. S. Sapatnekar, and I. N. Hajj, “Delay and Area Optimization for Discrete Gate Sizes under Double-Sided Timing Constraints,” in *Custom Integrated Cicruits Conference*. IEEE, May 1993, pp. 9.4.1–9.4.4.

[7] 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.

[8] J. V. Manoranjan and K. S. Stevens, “Implementation of Burst-Mode Controllers Using Relative Timing on FPGAs,” in *Southern Programmable Logic Conference*. IEEE, November 2014.

[9] Y. Xu and K. S. Stevens, “Automatic Synthesis of Computation Interference Constraints for Relative Timing,” in *26th International Conference on Computer Design*. IEEE, Oct. 2009, pp. 16–22.

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

[11] M. D. Riedel and J. Bruck, “Timing Analysis of Cyclic Combinational Circuits,” in *International Workshop on Logic and Synthesis*. IEEE, 2004, pp. 69–77.

[12] A. M. Lines, “Pipelined Asynchronous Circuits,” Master’s thesis, California Institute of Technology, Pasadena, CA, 1998.

[13] S. B. Furber, “A Small Compendium of 4-Phase Macropipeline Latch Control Circuits,” University of Manchester, Dept. of Computer Science, Technical Report v0.3, 17/01/99, 1999.