Qualifying Relative Timing Constraints for Asynchronous Circuits

Jotham Vaddaboina Manoranjan  Kenneth S. Stevens
University of Utah
Salt Lake City

International Symposium on Asynchronous Circuits and Systems

May, 2016
Abstract Statement

Employ signal path analysis to rank sets of relative timing constraints. Given a Solution Set \( set_0, set_1, ..., set_n \) for a Circuit \( ckt_i \):

\[
\begin{align*}
set_0 &= rtc_0^0, rtc_0^1, ..., rtc_0^n \\
set_1 &= rtc_1^0, rtc_1^1, ..., rtc_1^n \\
... &= rtc_n^0, rtc_n^1, ..., rtc_n^n \\
\end{align*}
\]

Find the best solution set!
Relative timing is a formalism that explicitly represents timing requirements. RT constraints are used to:
- make hazards unreachable
- guarantee functional correctness of circuit

Modular design capability

Successfully applied to ASIC and FPGA based designs

Process generation of advantage
Relative Timing

Formalism

\[ pod \mapsto poc_0 \prec poc_1 \]

The above constraints specifies that after the point-of-divergence \( pod \), point-of-convergence \( poc_0 \) must occur before \( poc_1 \).

maximum delay \((pod \text{ to } poc_0)\) < minimum delay \((pod \text{ to } poc_1)\)

Figure : Implementation with Delays
Applying the formalism

LEFTI = rst.LEFT
LEFT = lr.c1.'la. c2.lr.'la.LEFT
RIGHT = 'c1.'rr.'c2.ra.'rr.ra.RIGHT
SPEC = (LEFTI | LEFT | RIGHT) \ { c1,c2 }

Figure: A Burst-mode controller

Figure: CCS specification of the burst-mode controller
Circuit and Model

- Speed Independent vs Delay Insensitive
  - The circuit analysis employs delay insensitive models
  - Most accurate models
- ASICs vs FPGAs
  - Profound Impact on FPGA
  - FPGAs cannot employ Speed Independent models
- Signal delays are comparable to the logic delays on an FPGA
Controller Implementation on an FPGA

An FPGA based controller implementation is utilized to present the methodology.

Figure: Look-up-table based controller implementation
Fully Expressing the Controller

Figure: Controller with modeled forks
Given the circuit, what does an RT constraint look like?

\[ lr \uparrow \rightarrow rr0 \uparrow + m < y2 \downarrow \]  

Figure: Implementation with Delays
Relative Timing

**Figure:** Max Constraint Path

\[ lr \uparrow \quad \Rightarrow \quad rr0 \uparrow + m \quad \triangleleft \quad y2 \downarrow \quad \text{max-path: } \quad lr \uparrow \quad \Rightarrow \quad rr0 \uparrow \quad (2) \]

This is the early arrival path
Relative Timing

Figure: Min Constraint Path

\[ lr \uparrow \rightarrow rr0 \uparrow + m \prec y2 \downarrow; \; \text{min-path:} \; lr \uparrow \rightarrow y2 \downarrow \]  \hspace{1cm} (3)

This is the late arrival path
That was just one constraint!

RTC0: \( pod_0 \rightarrow poc_0 \prec poc_1 \)
That’s more like it!

RTC0: $pod_0 \mapsto poc_0 \prec poc_1$
RTC1: $pod_1 \mapsto poc_1 \prec poc_1$
RTC2: $pod_2 \mapsto poc_2 \prec poc_1$
RTC3: $pod_3 \mapsto poc_3 \prec poc_1$
RTC4: $pod_4 \mapsto poc_4 \prec poc_1$

........

RTCn: $pod_n \mapsto poc_n \prec poc_n$

Each circuit usually require multiple RT constraints to be satisfied. Together these are called a Set of RT constraints or an RTC set.
Each circuit can have more than one RTC set associated with it.

Each of these sets, when faithfully implemented guarantee functional correctness.

These sets are heuristically created using ARTIST.
- Given a circuit implementation and a formal design specification, ARTIST automatically generates sets of relative 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.
Qualifying RT Sets: Parameters

<table>
<thead>
<tr>
<th>Set0</th>
<th>Set1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set2</td>
<td>Set3</td>
</tr>
<tr>
<td>......</td>
<td></td>
</tr>
<tr>
<td>Setm</td>
<td>Setn</td>
</tr>
</tbody>
</table>
ARE ALL RELATIVE TIMING CONSTRAINT SETS BORN EQUAL?
ARE ALL RELATIVE TIMING CONSTRAINT SETS BORN EQUAL?

No!

- Circuits usually have more than one possible constraint sets - numerous “easy to implement constraints” would be preferable over a set with a few “hard to implement constraints”
- Each sets can have many constraints that interact with each other - conflicting timing requirements
- Constraint sets impact circuit performance - delays may be needed to force conformance

Qualitative analysis of constraint sets can enable better circuit design choices
Key parameters:

- Robustness - Larger margin = better constraint! - Shorter early path and longer late path
- Timing conflicts - Two sided timing constraints
  - RTC0: $pod \rightarrow poc_0 \prec poc_1$
  - RTC1: $pod \rightarrow poc_1 \prec poc_0$
- Conflicting requirements
Robustness of Constraints

$lr1 \rightarrow rr0 \prec y2$

RT inequality:

$lr1 + d2 + rr0 + m \leq lr1 + d2 + rr1 + d3 + y2$

A margin $m$ is incorporated
\[ lr1 + d2 + rr1 + m \leq lr1 + d2 + rr1 + d3 + y2 \]
\[ m = (lr1 + d2 + rr1 + d3 + y2) - (lr1 + d2 + rr1) \]
\[ m = (D_S + D_L + D_S + D_L + D_S) - (D_S + D_L + D_S) \]
\[ m = D_S + D_L \]

\( m \) represents inherent robustness
• higher the value to $m$ the better the robustness of the constraint
• Each constraint in the set is analyzed and the value of $m$
• worst-case value of $m$ in a set are used to qualify the constraint set
Analyzing Timing Conflicts

Possible competing timing requirements:

RTCA: $x \rightarrow y \prec z$
RTCB: $x \rightarrow w \prec y$

*RTCA* may be attempting to reduce delay in the path $x\rightarrow y$ while *RTCB* is trying to increase the path delay.
Interacting paths:

Early Path: \([A, X, Y, X, Y, Z]\)
Late Path: \([B, X, Y, Z]\)

- paths are viewed as sets
- 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.
Figure: Examples of various ways in which paths from different RTCs can overlap.

Strong conflicts can create competing timing requirements that cannot be met.
Weak conflicts may also create competing timing requirements, however due to non overlapping portions, they can always be met.
Figure: Various possible conflicts
A directed graph $G = (V, E)$ is utilized to represent two-sided timing constraints.

$V$ is the set of vertices. Each individual RTC is represented by a Vertex.

A conflict between two RTCs is represented by a directed edge connecting the two:

RTC0: $lr \rightarrow y0 \prec rr1$

RTC1: $lr \rightarrow y1 \prec la0$
RTC0: \( lr1 \rightarrow y0 \prec rr1 \)
RTC0: $lr \mapsto y0 \prec rr1$

RTC1: $lr \mapsto y1 \prec la0$
RTC0 places a min delay requirement between $lr_1$ and $rr_1$
RTC1 places a max delay between $lr_1$ and $y_1$
These constraints have overlapping paths
RTC0 places a min delay requirement between $lr1$ and $rr1$
RTC1 places a max delay between $lr1$ and $y1$
These constraints have overlapping paths

RTC0

RTC1
RTC0 places a min delay requirement between $lr_1$ and $rr_1$
RTC1 places a max delay between $lr_1$ and $y_1$
These constraints have overlapping paths
RTC0: \( pod \mapsto poc_0 \prec poc_1 \)
RTC1: \( pod \mapsto poc_1 \prec poc_0 \)
Two sided constraints on both paths that can never be simultaneously resolved
Analyzing Timing Conflicts

Figure: Cyclical Conflicts

Strong Conflicts are represented by solid lines, and weak by dotted lines.
Given a RTC set from a circuit:

- The minimum delay constrained path (early path) of an RTC may conflict with the maximum delay constrained (late path) of the other RT constraints within a set.
- The minimum delay constrained path (early path) of an RTC may conflict with the maximum delay constrained (late path) of the other RT constraints within a set.

Hence each RTC can be analyzed for cyclical dependencies between constraints.

Each constraint set is analyzed for Strong Cyclical Conflicts and Weak Cyclical Conflicts
With our analysis of constraint sets, we now have the following information for each set:

1. Inherent robustness ($m$)
2. Weak conflicts ($wc$)
3. Strong conflicts ($sc$)
4. Weak cyclical conflicts ($wcc$)
5. Strong cyclical conflicts ($scc$)
Constraint set metrics

With our analysis of constraint sets, we now have the following information for each set:

1. **Inherent robustness** \((m)\)
2. Weak conflicts \((wc)\)
3. **Strong conflicts** \((sc)\)
4. **Weak cyclical conflicts** \((wcc)\): With at least one strong conflict
5. **Strong cyclical conflicts** \((scc)\)
What else can we use this for?

Figure: Controller with modeled forks

- A SI based circuit is first modeled
- Extracted constraints sets are evaluated and the best set is picked
- The best constraint set is utilized and one single fork is added
- The extraction of RT constraints is rerun
RTC set0

rtc0 : lr $\mapsto$ y0 $\prec$ la1;
$(lr1 + d2 + rr1 + d3 + y0 + m \leq lr0 + d1 + la + D_E + lr0 + d1 + la1)$

rtc1 : lr1 $\mapsto$ y0 $\prec$ rr1;
$(lr1 + d2 + rr1 + d3 + y0 + m \leq lr1 + d2 + rr + D_E + ra + d3 + rr1)$

rtc2 : lr1 $\mapsto$ y2 $\prec$ y;
$(lr1 + d2 + rr1 + d3 + y2 + m \leq lr1 + d2 + rr + D_E + ra + d2 + rr1 + d2)$

rtc3 : lr $\mapsto$ y2 $\prec$ lr1;
$(lr1 + d2 + rr1 + d3 + y2 + m \leq lr0 + d1 + la + D_E + lr0 + d1 + la + D_E + lr1)$

Figure : Relative timing constraint set: set0
Another Set

rtc0 - rtc2 - same as rtc2 from Set0
rtc3 : lr \mapsto y_1 \prec lr_1;
(lr_1 + d_2 + rr_1 + d_3 + y_1 + m \leq lr_0 + d_1 + la + D_E + lr_0 + d_1 + la + D_E + lr_1)
rtc4 : lr \mapsto lr_1 \prec y_2;
(lr_1 + d_1 + la + D_E + lr_1 + d_1 + la + D_E + lr_1 + m \leq lr_1 + d_2 + rr_1 + d_3 + y_2)
rtc5 : lr \mapsto y_0 \prec rr_1;
(lr_1 + d_2 + rr + D_E + ra_1 + d_2 + rr_1 + d_3 + y_0 + m \leq
lr_0 + d_1 + la + D_E + lr_0 + d_1 + la + D_E + lr_1 + d_2 + rr_1)
rtc6 : rr_1 \mapsto y_2 \prec y;
(rr_1 + d_3 + y_2 + m \leq rr_1 + d_3 + y_1 + d_1 + la_1 + d_3)
rtc7 : lr \mapsto ra_1 \prec lr_1;
(lr_0 + d_1 + la + D_E + lr_0 + d_1 + D_E + lr_1 + d_2 + rr + D_E + ra_1 + m \leq
lr_1 + d_2 + rr + D_E + ra_1 + D_E + rr_1 + d_3 + y_1 + d_1 + la + D_E + lr_1)
rtc8 : lr \mapsto lr_1 \prec ra_1;
(lr_1 + d_2 + rr + D_E + ra_1 + D_E + rr_1 + d_3 + y_1 + d_1 + la + D_E + lr_1 + m \leq
lr_0 + d_1 + la + D_E + lr_0 + d_1 + D_E + lr_1 + d_2 + rr + D_E + ra_1)
Results and Implementation

Table: RTC set0 analysis

<table>
<thead>
<tr>
<th>Constraint#</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtc0</td>
<td>$1 \times D_S + 1 \times D_L$</td>
</tr>
<tr>
<td>rtc1</td>
<td>$1 \times D_S + 1 \times D_L$</td>
</tr>
<tr>
<td>rtc2</td>
<td>$2 \times D_S + 1 \times D_L$</td>
</tr>
<tr>
<td>rtc3</td>
<td>$2 \times D_S + 2 \times D_L$</td>
</tr>
<tr>
<td><strong>Worst-case m</strong></td>
<td>$1 \times D_S + 1 \times D_L$</td>
</tr>
<tr>
<td><strong>Average m</strong></td>
<td>$1.5 \times D_S + 1.3 \times D_L$</td>
</tr>
</tbody>
</table>

Number of Strong Conflicts: 0
Number of Strong Cycles: 0
Number of Weak Cycles: 0

Analysis was automated
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.
### Table: Controller implementation results on the FPGA

<table>
<thead>
<tr>
<th>Set</th>
<th>Worst-case ( m )</th>
<th>Average ( m )</th>
<th>Strong Conflicts</th>
<th>Cyclic Conflict</th>
<th>Max margin (ps)</th>
<th>Min margin (ps)</th>
<th>Ave margin (ps)</th>
<th>Constr. met?</th>
</tr>
</thead>
<tbody>
<tr>
<td>set0</td>
<td>( 1 \times D_S + 1 \times D_L )</td>
<td>( 1.5 \times D_S + 1.3 \times D_L )</td>
<td>0</td>
<td>0</td>
<td>661</td>
<td>212</td>
<td>410</td>
<td>Yes</td>
</tr>
<tr>
<td>set1</td>
<td>( 1 \times D_S + 1 \times D_L )</td>
<td>( 1.5 \times D_S + 1.3 \times D_L )</td>
<td>0</td>
<td>0</td>
<td>438</td>
<td>317</td>
<td>358</td>
<td>Yes</td>
</tr>
<tr>
<td>set2</td>
<td>( -3 \times D_S + -3 \times D_L )</td>
<td>( 0.7 \times D_S + 0.4 \times D_L )</td>
<td>11</td>
<td>1</td>
<td>334</td>
<td>-248</td>
<td>148</td>
<td>No</td>
</tr>
<tr>
<td>set3</td>
<td>( -3 \times D_S + -3 \times D_L )</td>
<td>( 0.9 \times D_S + 0.7 \times D_L )</td>
<td>9</td>
<td>0</td>
<td>474</td>
<td>-173</td>
<td>116</td>
<td>No</td>
</tr>
<tr>
<td>set4</td>
<td>( -3 \times D_S + -3 \times D_L )</td>
<td>( 0.9 \times D_S + 1 \times D_L )</td>
<td>7</td>
<td>0</td>
<td>532</td>
<td>-173</td>
<td>196</td>
<td>No</td>
</tr>
<tr>
<td>set5</td>
<td>( -3 \times D_S + -3 \times D_L )</td>
<td>( 0.9 \times D_S + 0.9 \times D_L )</td>
<td>7</td>
<td>0</td>
<td>532</td>
<td>-173</td>
<td>179</td>
<td>No</td>
</tr>
<tr>
<td>set6</td>
<td>( -3 \times D_S + -3 \times D_L )</td>
<td>( 1.1 \times D_S + 1 \times D_L )</td>
<td>7</td>
<td>0</td>
<td>535</td>
<td>-173</td>
<td>267</td>
<td>No</td>
</tr>
<tr>
<td>set7</td>
<td>( -3 \times D_S + -3 \times D_L )</td>
<td>( 1.2 \times D_S + 1 \times D_L )</td>
<td>7</td>
<td>0</td>
<td>769</td>
<td>-173</td>
<td>222</td>
<td>No</td>
</tr>
<tr>
<td>set8</td>
<td>( -3 \times D_S + -3 \times D_L )</td>
<td>( 1.1 \times D_S + 1 \times D_L )</td>
<td>7</td>
<td>0</td>
<td>769</td>
<td>-173</td>
<td>267</td>
<td>No</td>
</tr>
<tr>
<td>set9</td>
<td>( -3 \times D_S + -3 \times D_L )</td>
<td>( 1.3 \times D_S + 1.1 \times D_L )</td>
<td>7</td>
<td>0</td>
<td>665</td>
<td>-173</td>
<td>208</td>
<td>No</td>
</tr>
</tbody>
</table>
Table: RT generation runtime reduction using developed tool for FPGA controller implementation

<table>
<thead>
<tr>
<th></th>
<th>RT generation runtime (s)</th>
<th>Analysis tool runtime (s)</th>
<th>Total runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct DI model</td>
<td>7139.52</td>
<td>-</td>
<td>7139.52</td>
</tr>
<tr>
<td>Iterative model</td>
<td>8.01</td>
<td>5.85</td>
<td>13.86</td>
</tr>
<tr>
<td>Total runtime reduction</td>
<td>-</td>
<td>-</td>
<td>99.8%</td>
</tr>
</tbody>
</table>
### Table: RT generation runtime reduction for asynchronous controllers using developed tool

<table>
<thead>
<tr>
<th>Controller</th>
<th>Reference</th>
<th>Traditional runtime (s)</th>
<th>Iterative tool runtime (s)</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC_BM</td>
<td>[1]</td>
<td>7,139.52</td>
<td>13.86</td>
<td>99.8%</td>
</tr>
<tr>
<td>PCHB</td>
<td>[2]</td>
<td>3.56</td>
<td>0.53</td>
<td>85.1%</td>
</tr>
<tr>
<td>BRF1,LH1</td>
<td>[3]</td>
<td>10,342.71</td>
<td>297.62</td>
<td>97.2%</td>
</tr>
</tbody>
</table>


Conclusion

- A methodology to evaluate and rank constraint sets has been presented
- RT constraint robustness and timing conflicts are analyzed
- A methodology to identify cyclical timing conflicts is presented
- The methodology is used to optimized RTs constraint for DI models


The End