Symbolic timing analysis for the verification of asynchronous circuits

Robert Clarisó    Jordi Cortadella
Universitat Politècnica de Catalunya
Barcelona, Spain

28 January 2003  3rd ACiD-WG Workshop

Contents

■ Motivation and previous work
■ Timed Transition Systems
■ Abstract interpretation of TTS
■ Examples
Motivation

Obtain an automatic procedure of the form:

**Input:** asynchronous circuits with symbolic delays

**Output:** “The circuit is hazard-free if delays are buffer < not + or”

Contribution

- A symbolic timing analysis technique
  - Based on abstract interpretation [Cousot, POPL’77]
  - Circuits: modeled as Timed Transition Systems (TTS)
  - Timing analysis is approximate and conservative
  - Result: constraints on delays values that guarantee the absence of hazards
- Useable for other safety properties
- Works in examples with 15 symbols
Previous work

- Timing analysis of asynchronous circuits
  - The formalism is usually Timed Automata (TA)
  - Very high complexity!
  - Approaches:
    - Difference Bound Matrices
    - Decision Diagram Techniques (BDD, NDD, DDD, ...)

- Symbolic timing analysis
  - Even higher complexity!
  - Approaches: Presburger arithmetics
    - Verification of Timing Diagrams [Amon, DAC ’97]
    - Time Separation of Events [Amon, ASYNC ’99]

Timed Transition Systems

[Henzinger]

A Timed Transition System (TTS) is defined by
- A non-empty set of states $S$
- A non-empty alphabet of events $\Sigma$
- A transition relation $T \subseteq S \times \Sigma \times S$
- An initial state $s_I$
- Lower ($\delta$) and upper ($\Delta$) delay bounds for each event

<table>
<thead>
<tr>
<th>Event</th>
<th>$\delta$</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>$\beta$</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
Modeling circuits as TTS

The state graph of a circuit is a TTS

Hazard: firing an event of the TTS disables another event
Symbolic timing analysis

- Based on abstract interpretation
- Instead of computing the exact clock values $C$, compute an abstraction (upper approximation) of the clock values

upper $C \subseteq A(C)$
approximation $C = A(C)$ not required

- $TTS \equiv$ system of equations among the clock values
- The system of equations is solved by fixpoint
- Abstraction vs exact analysis
  - less precise
  - more efficient

Abstractions

Example of abstraction: convex polyhedra

“Instead of describing a set of assignments to clocks, describe the system of linear constraints that they satisfy”

\[
\begin{align*}
\{ & 0 \leq \text{clock}_x \leq \text{not} \} \\
\{ & 0 \leq \text{clock}_y \leq \text{or} \land \\
& 0 \leq \text{clock}_z \leq \text{buffer} \} \\
\{ & \text{clock}_z = \text{or} \land \\
& 0 \leq \text{clock}_z = \text{or} \leq \text{buffer} \}
\end{align*}
\]
System of equations

- States
  - Precondition: \( \text{Pre}(s) = \cup \text{Post}(s \rightarrow s) \)
  - Postcondition: \( \text{Post}(s) = \text{Pre}(s) \)

- Transitions
  - Precondition: \( \text{Pre}(t) = \text{Post}(t) \)
  - Postcondition: \( \text{Post}(t) = \text{transfer}(\text{Pre}(t)) \)

\[ \begin{array}{c}
\text{a} \\
\alpha \\
\beta \\
\text{c} \rightarrow \text{d} \end{array} \]

System of equations

- States
  - Precondition: \( \text{Pre}(s) = \cup \text{Post}(s \rightarrow s) \)
  - Postcondition: \( \text{Post}(s) = \text{Pre}(s) \)

- Transitions
  - Precondition: \( \text{Pre}(t) = \text{Post}(t) \)
  - Postcondition: \( \text{Post}(t) = \text{transfer}(\text{Pre}(t)) \)

\[ \begin{array}{c}
\text{a} \\
\alpha \\
\beta \\
\text{c} \rightarrow \text{d} \end{array} \]

\[ \begin{array}{c}
\text{Pre}(\alpha) = \text{Post}(b) \\
\text{Post}(\alpha) = \text{transfer}(\text{Pre}(\alpha)) \end{array} \]
System of equations

- **States**
  - Precondition: \( \text{Pre}(s) = \text{U Post}(*\rightarrow s) \)
  - Postcondition: \( \text{Post}(s) = \text{Pre}(s) \)

- **Transitions**
  - Precondition: \( \text{Pre}(t) = \text{Post}(t) \)
  - Postcondition: \( \text{Post}(t) = \text{transfer}(\text{Pre}(t)) \)

\[
\begin{align*}
\text{Pre}(b) &= \text{Post}(\alpha) \cup \text{Post}(\beta) \\
\text{Post}(b) &= \text{Pre}(b) \\
\text{Pre}(c) &= \text{Pre}(c) \lor (\text{Post}(\alpha) \cup \text{Post}(\beta))
\end{align*}
\]
Symbolic transfer function

Post(t) = transfer( Pre(t) )

- transfer updates the clocks when an event is fired
- an event firing must satisfy lower and upper delay bounds

\[
\begin{align*}
\text{clock}_\alpha & := ? \\
\text{clock}_\beta & := 0
\end{align*}
\]

\[
\begin{align*}
\text{clock}_\alpha & := 0 \\
\text{clock}_\beta & := \text{clock}_\beta + \text{step}
\end{align*}
\]

Symbolic transfer function

Post(t) = transfer( Pre(t) )

- transfer “advances” the clocks when an event is fired
- an event firing must satisfy lower and upper delay bounds

\[
\begin{align*}
\text{transfer}_a (P) \\
P & := P \land ( \text{step} \geq 0 ) \\
P & := P \land ( \text{clock}_\alpha + \text{step} = \text{delay}_\alpha ) \\
P & := P \land ( \text{clock}_\beta + \text{step} \leq \text{delay}_\beta ) \\
P \left[ \text{clock}_\alpha := 0 \right] \\
P \left[ \text{clock}_\beta := \text{clock}_\beta + \text{step} \right] \\
P \left[ \text{clock}_\alpha := ? \right] \\
P \left[ \text{step} := ? \right]
\end{align*}
\]
Symbolic transfer function

\[ \text{Post}(t) = \text{transfer}( \text{Pre}(t) ) \]

- transfer "advances" the clocks when an event is fired
- an event firing must satisfy lower and upper delay bounds

**Pre(\(\alpha\))** : \{ clock\(\alpha\) = 0 \(\land\) clock\(\beta\) = 0 \}

**Post(\(\alpha\))** : \{ clock\(\beta\) = delay\(\alpha\) \(\land\) delay\(\alpha\) \(\leq\) delay\(\beta\) \}

Convex polyhedra operations

- **Union (convex hull)**
- **Widening**
- **Linear assignment**
- **Existential (Fourier-Motzkin)**
Overall flow

1. Compute the state graph (TTS) of the circuit
2. Define the equations that relate Pre/Post
3. Solve the system of equations
   - Pre/Post: abstracted as a convex polyhedron
   - Initialize all polyhedra as $\emptyset$
   - Apply equations iteratively
   - Stop when $\text{solution}_{k+1} \subseteq \text{solution}_k$
4. Hazard analysis
   - Find hazard transitions in the TTS
   - Get the Post of the hazard & undefine clock symbols
   - Result: constraints on symbolic delay bounds

Using results of timing analysis

Post(x-) =
   \{ \text{clock}_y = 0 \land \text{buffer} \geq \text{not + or} \}
\Rightarrow \{ \text{buffer} \geq \text{not + or} \}
\Rightarrow \{ \text{buffer} < \text{not + or} \}
**Burst-mode controller**

- No hazards if \( g_4 + g_5 < c + g_1 + g_6 \)
- \( g_1 < c + g_2 + g_7 \)
- Verification time: 34 seconds

**Asynchronous pipeline**

- **Safety property**: stage1 must be empty when the environment adds new elements

\[ \text{in} > \text{out} \land (\text{in} > \text{stage1}) \land \ldots \land (\text{in} > \text{stageN}) \]

\[ \text{in} > \max (\text{out}, \text{stage1}, \ldots, \text{stageN}) \]
Asynchronous pipeline

\[ \text{IN} \xrightarrow{\text{stage1}} \text{req} \xrightarrow{\text{ack}} \text{stage2} \xrightarrow{\text{req}} \text{ack} \xrightarrow{\text{...}} \text{req} \xrightarrow{\text{ack}} \text{stageN} \xrightarrow{\text{OUT}} \]

<table>
<thead>
<tr>
<th>N</th>
<th>States</th>
<th>Fully symbolic</th>
<th>Environment + 1 stage</th>
<th>Constant delays</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td># sym</td>
<td>time</td>
<td># sym</td>
</tr>
<tr>
<td>2</td>
<td>48</td>
<td>8</td>
<td>2.7</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>144</td>
<td>10</td>
<td>24.2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>432</td>
<td>12</td>
<td>135</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1296</td>
<td>14</td>
<td>1073</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>3888</td>
<td>16</td>
<td>—</td>
<td>4</td>
</tr>
</tbody>
</table>

Future work

- Automatic simplification of result conditions
- Symbolic representation of states
  - Avoid explicit generation of the state graph
  - Alternative representations for convex polyhedra
  - Decision Diagram techniques?
- Application to other formalisms
  - Timed Automata
  - Hybrid Systems
Conclusions

- Symbolic timing analysis is capable of verifying wider classes of concurrent systems

- Based on abstract interpretation
  - clock regions represented as *convex polyhedra*
  - provides *sufficient* constraints for correctness
  - calculates an *approximate* timed state space
  - no false positives

- Applied to hazard analysis, but valid for other safety properties