Reasoning about Timed Systems
Using Boolean Methods

Sanjit A. Seshia
EECS, UC Berkeley

Joint work with
Randal E. Bryant (CMU)
Kenneth S. Stevens (Intel, now U. Utah)
Timed System

A system whose correctness depends not only on its **functionality** (what results it generates), but also on its **timeliness** (the time at which results are generated).
Real-Time Embedded Systems
Self-Timed Circuits

HINTON et al.: CMOS IA-32 PROCESSOR WITH 4-GHz INTEGER EXECUTION UNIT

Fig. 9. ALU design in global STP: a typical bit.
Modeling & Verification

Verify model

Model

Timed System
Challenges with Timed Systems

- State has 2 components:
  - Boolean variables ($V$): model discrete state
  - Real-valued variables ($X$): measure real time

- Infinitely-many states
  - Has a finite representation (regions graph)
  - But grows worse than $|X|^{[X]}$
  - Verification is hard!
Modeling & Verification

Verify model

Model

Timed System

Model Checking

Timed Automaton

Self-Timed Circuit
Message of This Talk: Leverage Boolean Methods

- **Modeling**
  - Use Boolean variables to model timing, where possible

- **Verification**
  - Use symbolic Boolean representations and algorithms operating on them
    - Binary Decision Diagrams (BDDs), Boolean satisfiability solvers (SAT)

- **Why?**
  - Systems have complex Boolean behavior anyway
  - Great progress made in finite-state model checking, SAT solving, etc. over last 15 years
Talk Outline

- Motivating Problem: Verifying Self-Timed Circuits
- Generalized Relative Timing
- Circuits $\rightarrow$ Timed Automata
- Model Checking Timed Automata
- Case Studies
- Future Directions & Related Research
Self-Timed (Asynchronous) Circuits

Many design styles use timing assumptions

- Relative Timing: [Stevens et al. ASYNC’99, TVLSI’03]
  - Circuit behavior constrained by relative ordering of signal transitions
    - \( u \uparrow \prec \downarrow v \uparrow \)
Relative Timing (RT) Verification Methodology: 2 Steps

1. Check circuit functionality *under timing assumptions*
   - Search the constrained state space
   - Model checking

2. Verify timing assumptions themselves
   - Size circuit path delays appropriately
   - Static timing analysis
Pros and Cons of RT

- **Advantages:**
  - Applies to many design styles
  - Incremental addition of timing constraints
  - No conservatively set min-max delays

- **Disadvantages:**
  - Cannot express metric timing
  - More work to be done on verification
    - Scaling up
    - Validating timing constraints themselves
Our Contributions

Generalized RT
  – Can express some metric timing

Applied Fully Symbolic Verification Techniques
  – Model circuits using timed automata
    ■ Metric timing modeled using real-valued variables
    ■ Non-metric with Booleans

Performed Case Studies
  – Including Global STP circuit (published version of Pentium-4 ALU ckt.)
Motivating Problem: Verifying Self-Timed Circuits

Generalized Relative Timing

- Circuits → Timed Automata
- Model Checking Timed Automata
- Case Studies
- Future Directions & Related Research
Generalizing Relative Timing

Relative Timing

- Delay Independent
- Burst Mode
- Gate-level Metric Timing
Circuit Model

- Variables (signals): $v_1, v_2, \ldots, v_n$

- Events (signal transitions): $e_i$ is $v_i \uparrow$ or $v_i \downarrow$

- Rules
  - $\mathcal{E}_i (v_1, v_2, \ldots, v_n) \rightarrow e_i$

- Timing Constraints
Generalized Relative Timing (GRT) Constraint

- $\Delta(e_i, e_j)$: Time between $e_j$ and previous occurrence of $e_i$

- Form of GRT constraint:

$$\Delta(e_i, e_j) \leq \Delta(e_i', e_k) + d$$
Special Case: Common Point-of-Divergence (PoD)

- **PoD constraint:**
  \[ \Delta(e_i, e_j) \leq \Delta(e_i, e_k) \]

- **Written as:**
  \[ e_i \rightarrow e_j \prec e_k \]

- **An RT constraint traced back to its source**
Example: Point-of-Divergence (PoD) Constraint

\[ c \uparrow \rightarrow ac \uparrow \prec b \downarrow \]

Diagram:

- Nodes: a, b, c
- Edges: a\rightarrow ab\rightarrow ac\rightarrow bc\rightarrow c
Example: Metric Timing

\[ \Delta(\text{data}_\text{in}↑, \text{data}_\text{in}\_\text{aux}↑) \leq \Delta(\text{enable}↑, \text{trigger}↑) \]
Do We Need Metric Timing?

- Useful for *modular specification* of timing constraints
- Also when delays are explicitly used
Verifying Generalized Relative Timing Constraints

- Use static timing analysis to compute min-max path delays

- To verify:

  \[ \Delta(e_i, e_j) \leq \Delta(e'_i, e_k) + d \]

  We verify that:

  \[ \text{max-delay}(e_i \sim e_j) \leq \text{min-delay}(e'_i \sim e_k) + d \]
Talk Outline

- Motivating Problem: Verifying Self-Timed Circuits

- Generalized Relative Timing

  - Circuits $\rightarrow$ Timed Automata

    - Model Checking Timed Automata

    - Case Studies

    - Future Directions & Related Research
Modeling Timed Circuits

- Need to model:
  **Rules** ("Boolean" behavior) and **Timing**

- **Our formalism**: **Timed Automata** [Alur & Dill, ’90]
  - Generalization of finite automata
  - **State variables:**
    - Boolean (circuit signals)
    - Real-valued timers or "clocks" (impose timing constraints)
      - Operations: (1) compare with constant, (2) reset to zero

- **We model non-metric timing with Booleans**
Enforcing Timing with Booleans

\[ c \uparrow \rightarrow ac \uparrow \prec b \downarrow \]

1. \( c \uparrow \) sets a bit

2. \( ac \uparrow \) resets it

3. \( b \downarrow \) cannot occur while the bit is set
Enforcing Timing with Timer Variables

\[ \Delta(\text{data\_in}↑, \text{data\_in\_aux}↑) \leq \Delta(\text{enable}↑, \text{trigger}↑) \]
Enforcing Timing with Timer Variables

\[
\Delta(\text{data\_in}^\uparrow, \text{data\_in\_aux}^\uparrow) \leq \Delta(\text{enable}^\uparrow, \text{trigger}^\uparrow)
\]

- \text{data\_in}^\uparrow \text{ sets } x_1 \text{ to } 0
- \text{data\_in\_aux}^\uparrow \text{ must occur while } x_1 \leq c
- \text{enable}^\uparrow \text{ sets } x_2 \text{ to } 0
- \text{trigger}^\uparrow \text{ can only occur if } x_2 \geq c

\[ c \text{ determined just as in other metric timing styles} \]
Booleans vs. Timers

- Most timing constraints tend to be PoD
- So few real-valued timer variables used in practice
Talk Outline

- Motivating Problem: Verifying Self-Timed Circuits
- Generalized Relative Timing
- Circuits $\rightarrow$ Timed Automata
  - Model Checking Timed Automata
- Case Studies
- Future Directions & Related Research
State

- **Boolean part: assignment to signals**
  
  \[ v_1 = 0, \ v_2 = 1, \ v_3 = 0, \ldots \]

- **Real-valued part: relation between timers**
  
  \[ x_1 \geq 0 \land x_2 \geq 0 \land x_1 \geq x_2 \]

  symbolic representation
Symbolic Model Checking of Timed Automata

Symbolically represent sets of signal assignments with corresponding relations between timers

\[ v_1 \lor v_2 \land x_1 \geq 0 \land x_2 \geq 0 \land x_1 \geq x_2 \]
Our Approach to Fully Symbolic Model Checking

- Based on algorithm given by Henzinger et al. (1994)
- Core model checking operations
  - Image computation
  - Quantifier elimination in quantified difference logic
  - Termination check
  - Satisfiability checking of difference logic

- Our Approach: Use Boolean encodings
  - Quantified difference logic
  - Quantified Boolean logic
  - Difference logic
  - Boolean logic
  - Use BDDs, SAT solvers
Example: Termination Check

- Have we seen all reachable states of the systems?

- Satisfiability solving in Difference Logic
Solving Difference Logic via SAT

\[ x \geq y \land y \geq z \land z \geq x+1 \]

\[ e_1 \land e_2 \land e_3 \]

\[ e_1 \land e_2 \Rightarrow \neg e_3 \]

Overall Boolean Encoding

Transitivity Constraint

\[ e_1 \quad x \geq y \]
\[ e_2 \quad y \geq z \]
\[ e_3 \quad z \geq x+1 \]
A More Realistic Situation

\[ x \geq y \land y \geq z \land z \geq x+1 \land \ldots \] is a term in the SOP (DNF)
Talk Outline

- Motivating Problem: Verifying Self-Timed Circuits

- Generalized Relative Timing

- Circuits $\rightarrow$ Timed Automata

- Model Checking Timed Automata

- Case Studies

- Future Directions & Related Research
Case Studies

- Global STP Circuit
  - Self-resetting domino ckt. in Pentium-4 ALU
  - Analyzed published ckt. [Hinton et al., JSSC’01]

- GasP FIFO Control [Sutherland & Fairbanks, ASYNC’01]

- STAPL Left-Right Buffer [Nystrom & Martin, ’02]

- STARI [Greenstreet, ’93]
Footed and Unfooted Domino Inverters
Global STP Circuit
(simplest version at gate-level)
Global STP Circuit: Sample Constraint

\[ \text{res} \rightarrow \text{ck} \]

\[ \text{ck} \uparrow \rightarrow \text{ck} \downarrow \leftarrow \text{res} \downarrow \]
Global STP Circuit: An Error

We want: red < blue
7 transitions < 5 transitions
Comparison with ATACS

- Model checking for absence of short-circuits

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Number of Signals</th>
<th>Time for our model checker, TMV (in sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global STP</td>
<td>28</td>
<td>66.32</td>
</tr>
<tr>
<td>GasP-10 stages</td>
<td>60</td>
<td>26.10</td>
</tr>
<tr>
<td>STAPL-3 stages</td>
<td>30</td>
<td>278.05</td>
</tr>
</tbody>
</table>

ATACS did not finish within 3600 sec. on any
Comparison with ATACS on STARI

![Graph showing comparison between TMV-POD, ATACS, and TMV-Metric with respect to total time and timeout for varying number of STARI Stages.](image)
Related Work

- **Modeling**
  - Gate-level Metric Timing
    - Timed Petri Nets, TEL, ... [Myers, Yoned, et al.]
  - Chain Constraints [Negulescu & Peeters]
  - Relative Timing [Stevens et al.]
    - Lazy transition systems [Pena et al.]
  - Symbolic Gate Delays [Clariso & Cortadella]

- **Verification**
  - For circuits, mostly restricted to just symbolic techniques [e.g., ATACS]
Talk Outline

- Motivating Problem: Verifying Self-Timed Circuits
- Generalized Relative Timing
- Circuits → Timed Automata
- Model Checking Timed Automata
- Case Studies
- Future Directions & Related Research
Summary

- Leverage Boolean Methods for Timed Systems
  - Modeling: *generalized relative timing*
  - Verification: *fully symbolic model checking*
    - Using BDDs, SAT

- Demonstrated Application: Modeling and Verifying Self-Timed Circuits
Future Directions: Model Generation

- Model
- Timed System

Main Challenge:
**Automatic generation of timing constraints**

Idea: Machine learning from simulated runs (successful and failing)
Future Directions: New Applications

- Distributed Real-time Embedded Systems
  - E.g., sensor networks
  - Operate asynchronously
  - Lots of concurrency
  - Timeliness important

- Will generalized relative timing work for this application?
Related Research Project

- **UCLID**
  - Modeling & Verifying Infinite-State Systems
  - Focus: Integer arithmetic, Data Structures (arrays, memories, queues, etc.), Bit-vector operations, ...
  - Applications: Program verification, Processor verification, Analyzing security properties
    - E.g., detecting if a piece of code exhibits malicious behavior (worm/virus)

- Also based on Boolean Methods
  - Problems in first-order logic translated to SAT

- Programming Systems seminar, Oct. 24 ’05
More information at
http://www.eecs.berkeley.edu/~sseshia/research.html

Thank you!