Formal verification of a complex timed circuit: IPCMOS

Marco A. Peña
Jordi Cortadella
Enric Pastor
Alexander Smirnov
Universitat Politècnica de Catalunya

Overview

GOAL: Formal verification of the IPCMOS architecture

IPCMOS pipelines ↓

IPCMOS stage

Abstractions + Assume Guarantee + Induction

Relative Timing based Verification - TranSyT

ASYNC 2000
Outline

- Formal verification with Relative Timing
- IPCMOS architecture
- Verification of IPCMOS pipelines
  - Strategy
  - Abstractions
  - Assume-Guarantee
  - Results
- Details on the verification of a single stage pipeline
- Conclusions

Formal verification with Relative Timing
Verification approach: main features

- Approach published in ASYNC 2000
- Iterative incremental refinement of the untimed state space by:
  - Off-line timing analysis on small acyclic graphs, and
  - Incorporation of Relative Timing constraints
- Applicable to Timed Transition Systems (TS + delay bounds), with any type of causality relations
- Verification of temporal safety properties
- BDD-based symbolic representation: large untimed state spaces
- Backannotation: sufficient relative timing constraints for correctness are reported

Verification approach: system model

Timed Transition System (Manna, Pnueli)

- Transition System
- Min/Max Delays
Verification approach: system model

**Timed Transition System**
(Manna, Pnueli)

- Transition System
- Min/Max Delays

\[ \delta(a) \in [1,2] \]
\[ \delta(b) \in [1,2] \]
\[ \delta(c) \in [2.5,3] \]
\[ \delta(g) \in [0.5,0.5] \]
\[ \delta(d,x,y) \in [0,\infty) \]

Verification approach: system model

**Lazy Transition System**
(Cortadella, et. al.)

- Transition System
- Relative Timing relations

\[ \delta(a) \in [1,2] \]
\[ \delta(b) \in [1,2] \]
\[ \delta(c) \in [2.5,3] \]
\[ \delta(g) \in [0.5,0.5] \]
\[ \delta(d,x,y) \in [0,\infty) \]
Verification approach

Symbolic state space exploration and failure detection

- Failure trace
- Event structure

Failure states

- Timing analysis
- Composition
Verification approach

- Failure trace
- Event structure

- Timing analysis
- Composition

Verification approach: flow

TTS  TS  + delay

LzTS

Property

Failure search

Failure Trace

Timing consistent?

yes

Counter example

no

Causality extraction

CES  Timing analysis  LzCES

Timing constraint composition
IPCMOS architecture

General IPCMOS architecture

- **Pulse-based** asynchronous clocking technique for large devices operating at GHz frequencies
- Block-level interlocking scheme ⇒ scalable
- Schuster, et al. ISSCC 2000
Linear IPCMOS pipeline architecture

- **VALID**: worst-case performance through a block ⇒ data is available to the next block
- **ACK**: data received by the next block
- Performance up to 4GHz
- Correctness depends on pulse widths

---

Two-stage pipeline at work
Two-stage pipeline at work

Critical pulse width

ACK depends on next stage
Two-stage pipeline at work

VALID depends on next stage

Pulse-based interface

2-phase handshake between stages
Stage building blocks

Strobe circuit: general

- Capture negative pulses at $\text{VALID}_i$ and produce a positive pulse at $\text{ACK}$ and a negative pulse at $\text{CLKE}$
Strobe circuit

- Capture a negative pulse at $\text{VALID}$ from the previous stage and produce a positive pulse at $\text{ACK}$ to the next stage and a negative pulse at $\text{CLKE}$ to the functional unit.

Strobe circuit: detail

- Transition relations:
  - $\text{En}(Y^+): \neg Y \cdot \neg Z$
  - $\text{En}(Y^-): Y \cdot \text{ACK}$
  - . . .

- Failure conditions:
  - Shortcut at $Y: \neg Z \cdot \text{ACK}$
  - Shortcut at $\text{Vint}: \neg \text{VALID} \cdot Y \cdot \neg \text{CLKE}$
Reset circuit: general

- Capture positive pulses at \text{ACK}_i and produce a positive pulse at \text{CLKR} and a negative pulse at \text{CLKRN}

Reset circuit

- Capture a positive pulse at \text{ACK} from the next stage and produce a positive pulse at \text{CLKR} and a negative pulse at \text{CLKRN} to reset both the strobe and valid circuits
Reset circuit: detail

- Capture a positive pulse at ACK from the next stage and produce a positive pulse at CLKR and a negative pulse at CLKRN to reset both the strobe and valid circuits.

Valid circuit

- Capture negative pulses at CLKE and produce a negative pulse at VALID. The pulse is reset by a positive pulse at CLKR.
- Delay after the inverter depends on the worst-case delay of the functional unit controlled by the stage.
Verification of IPCMOS pipelines

- Assuming data-path is correct, the pipeline is correct (spec.):
  \[ S = \text{“Every data fed into the pipeline is acknowledged once and only once at every stage”} \]
- \( S \) is modeled by a deadlock condition plus correctness of CMOS circuits: no short-circuits, etc.
- Correctness regardless of the length of the pipeline
  \[ \text{IN} \parallel I_1 \parallel \ldots \parallel I_n \parallel \text{OUT} \leq S, \ n \geq 1 \]
Assume-Guarantee verification

- Pnueli 1984, Clarke et al. 1989, etc.
- Abstractions to overcome complexity: preserve the input/output behavior and the properties of interest
- Assume the abstractions are correct
- Prove that the abstractions are correct to guarantee a sound analysis

Verification strategy

- Key observation:
  - Pulse-based communication only at the extremes
  - Internal communication is time-independent, i.e. 2-phase handshaking
- Allows:
  - Untimed abstractions
  - assume-guarantee
Verification strategy

- \( A_{in} \) and \( A_{out} \) are untimed abstractions that hide the pulse-based behavior.
- **Assume**: pose verification in terms of: \( A_{in} \parallel A_{out} \leq S \)
- **Guarantee** soundness of the abstractions:
  \[
  \text{IN} \parallel I_1 \parallel \ldots \parallel I_n \parallel \text{OUT} \leq A_{in} \parallel A_{out}
  \]
- Prove correctness of a one-stage pipeline

Abstractions
Assume-guarantee strategy

- Assume: $A_{\text{in}} \parallel A_{\text{out}} \leq S$
- Guarantee correctness of $A_{\text{out}}$
- Guarantee correctness of $A_{\text{in}}$
- Guarantee correctness of $A_{\text{in}}$ (induction)
- Guarantee correctness of 1-stage: $\text{IN} \parallel I \parallel \text{OUT} \leq S$

Assume-guarantee strategy

- Assume: $A_{\text{in}} \parallel A_{\text{out}} \leq S$
- Straightforward
- Few seconds of CPU time
**Assume-guarantee**

- Guarantee correctness of $A_{out}$:
  \[ A_{in} \parallel I \parallel OUT \leq A_{in} \parallel A_{out} \]
- Check that any output produced by $I \parallel OUT$ can also be produced by $A_{out}$ at the same time
- 28 minutes of CPU time

---

**Assume-guarantee**

- Guarantee correctness of $A_{in}$ with one stage:
  \[ IN \parallel I \parallel A_{out} \leq A_{in} \parallel A_{out} \]
- Check that any output produced by $IN \parallel I$ can also be produced by $A_{in}$ at the same time
- 9 minutes of CPU time
Assume-guarantee

- Guarantee $A_{in}$ is a *behavioral fixed point*:
  \[ A_{in} \parallel I \parallel A_{out} \leq A_{in} \parallel A_{out} \]
- 10 minutes of CPU time
- By induction, no matter how long the pipeline is, $A_{in}$ can be used as a correct abstraction

Assume-guarantee

- Guarantee correctness of a 1-stage pipeline:
  \[ \text{IN} \parallel I \parallel \text{OUT} \leq S \]
- 35 minutes of CPU time
- Most complex verification due to pulse-based environment
### Assume-guarantee: results

<table>
<thead>
<tr>
<th>Condition</th>
<th>CPU time</th>
<th>Refinements</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{in} \parallel A_{out} \leq S$</td>
<td>1m</td>
<td>--</td>
</tr>
<tr>
<td>$A_{in} \parallel I \parallel OUT \leq A_{in} \parallel A_{out}$</td>
<td>28m</td>
<td>7</td>
</tr>
<tr>
<td>$IN \parallel I \parallel A_{out} \leq A_{in} \parallel A_{out}$</td>
<td>9m</td>
<td>3</td>
</tr>
<tr>
<td>$A_{in} \parallel I \parallel A_{out} \leq A_{in} \parallel A_{out}$</td>
<td>10m</td>
<td>3</td>
</tr>
<tr>
<td>$IN \parallel I \parallel OUT \leq S$</td>
<td>35m</td>
<td>40</td>
</tr>
</tbody>
</table>

Details on the verification of a single stage pipeline
Environment and failure conditions

- Most aggressive environment: pulse-based
- Stage described at transistor level (32 transistors)
- Fail conditions:
  - Conformance of the interface
  - Avoidance of short-circuits
  - Signal persistency

A verification iteration

Failure trace. Is it time-feasible?
A verification iteration

Early Z- while Y still being discharged!

A verification iteration

ES capturing causality
A verification iteration

Off-line timing analysis

Sufficient minimum width of VALID pulse

A verification iteration

Reduced Timed ES
Relative timing constraints

Sufficient minimum width of VALID pulse
Conclusions

- RT-based verification: combine absolute timing (for analysis) with relative timing (for state space calculation)
- RT crucial to prove correctness of a complex timed system at transistor level
  - Relevant feature: back-annotation of sufficient RT-constraints that guarantee correct operation
  - Abstractions, assume-guarantee reasoning and induction to overcome complexity: but still manual

- Future work:
  - Reduce BDD size or try other data structures
  - Automate abstractions, AG-reasoning, etc.