

An Update on Self-Timed Circuit Verification

Cuong Chau

ckcuong@cs.utexas.edu

Department of Computer Science
The University of Texas at Austin

ACL2 Seminar Talk

September 21, 2018

Outline

- 1 Progress
- 2 Verification Flow
- 3 Arbitrated Merge
- 4 Conclusion

1 Progress

2 Verification Flow

3 Arbitrated Merge

4 Conclusion

HVC-2017 paper [Chau et al.:2017]: Modeled and verified a **32-bit self-timed serial adder**.

- Proved the correctness of circuit behavior under a condition that the circuit behavior follows an **explicitly declared interleaving set**.
- Our reasoning method **explored all declared interleavings, including ones internal to submodules**.
- Imposed a design restriction **preventing a module from communicating with other modules** until it becomes quiescent after finishing all of its internal processing.

ASYNC-2018 paper [Chau et al.:2018]: Modeled and verified **data-loop-free** self-timed circuits.

- Proved the correctness of circuit behavior without specifying any interleaving.
- Developed a new hierarchical verification method that avoids exploring the internal structures as well as operational interleavings of verified submodules.
- Avoided imposing the design restriction presented in the HVC-2017 paper.

Spring 2018: Modeled and verified **iterative** self-timed circuits that compute the **greatest-common-divisors (GCDs)** of their two input operands.

- Implemented **circuit generators** that mechanically generate circuit descriptions of **arbitrary data-bus width**, and verified such parameterized circuit generators.
- Showed that the verification method introduced in the ASYNC-2018 paper can be applied to iterative circuits as well.

Summer 2018: Modeled and verified self-timed circuits performing **arbitrated merge** operations.

- Formalized an **arbitrated merge joint** that provides mutually exclusive accesses to its output link from its two input links.
- Developed strategies for formalizing the relationship between the input and output sequences for modules containing such arbitrated merge joints.
- Developed a library that supports reasoning about the **membership relation** and the **interleaving operation**.

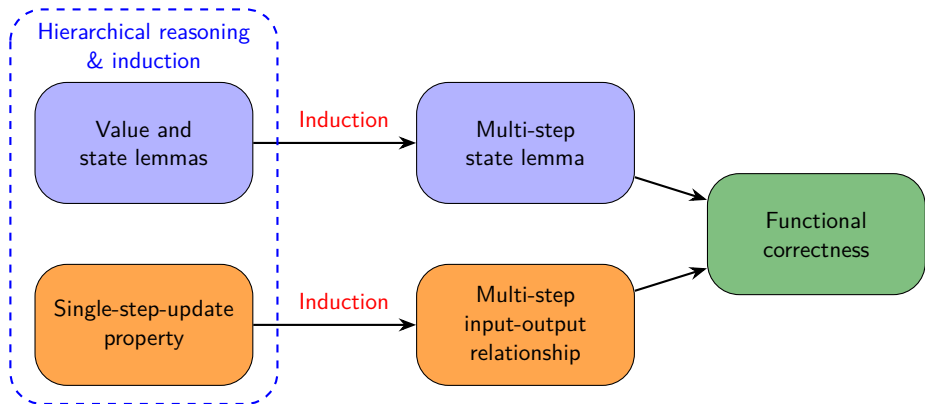
Fall 2018:

- Model and verify a **parameterized self-timed serial adder model** using our new approach.
 - Formalize **shift register** models based on the link-joint paradigm.
 - Re-design the serial adder model with the new shift register models. The new design should avoid imposing the design restrictions introduced in the HVC-2017 paper.
- Further demonstrate the **compositionality** of our new approach by proving the functional correctness of a GCD circuit model that contains the serial adders as submodules.

Outline

- 1 Progress
- 2 Verification Flow**
- 3 Arbitrated Merge
- 4 Conclusion

Verification Flow



Value and State Lemmas

Value lemma: characterize the module's outputs

$se(\text{module-name}, \text{inputs}, st, \text{netlist}) = \text{outputs}(\text{inputs}, st)$

State lemma: characterize the module's next state

$de(\text{module-name}, \text{inputs}, st, \text{netlist}) = \text{step}(\text{inputs}, st)$

outputs and step are **hierarchically** defined as **symbolic, four-valued expressions** that specify the module's outputs and next state, respectively.

Multi-Step State Lemma

Characterize the module's final state after an n -step execution.

$$de\text{-}n(\textit{module-name}, \textit{inputs-seq}, \textit{st}, \textit{netlist}, n) = \textit{run}(\textit{inputs-seq}, \textit{st}, n)$$

Multi-Step State Lemma

Characterize the module's final state after an n -step execution.

$de\text{-}n(\text{module-name}, \text{inputs-seq}, st, \text{netlist}, n) = \text{run}(\text{inputs-seq}, st, n)$

$\text{run}(\text{inputs-seq}, st, n) :=$

if ($n \leq 0$) st

else

$\text{run}(\text{rest}(\text{inputs-seq}),$
 $\text{step}(\text{first}(\text{inputs-seq}), st),$
 $n - 1)$

Single-Step-Update Property

Specify the input-output relationship after one execution step.

Introduce an **extraction function** for each self-timed module, $extract(st)$, that returns a sequence of values computed from state st that may ultimately appear in the output sequence.

Applying $extract$ to $step$ will compute the one-step update on the output sequence given the current inputs and current state. Note that $extract$ and $step$ are defined **hierarchically**.

Single-step-update property:

$$extract(step(inputs, st)) = extracted-step(inputs, st)$$

where $extracted-step$ is the specification for the one-step update on the output sequence.

Single-Step-Update Property

Let *in-act* and *out-act* denote the **communication signals** at the input and output ports respectively (Assume that the corresponding module has one input and one output ports).

$$\text{extracted-step}(inputs, st) := \begin{cases} \text{extract}(st), & \text{if } in\text{-act} = nil \wedge out\text{-act} = nil \\ [op(inputs.data)] ++ \text{extract}(st), & \text{if } in\text{-act} = t \wedge out\text{-act} = nil \\ \text{remove-last}(\text{extract}(st)), & \text{if } in\text{-act} = nil \wedge out\text{-act} = t \\ [op(inputs.data)] ++ \text{remove-last}(\text{extract}(st)), & \text{otherwise} \end{cases}$$

where

- $++$ is the concatenation operator;
- $\text{remove-last}(l)$ returns list l except for its last element; and
- op is the **functional specification** for the module.

Multi-Step Input-Output Relationship

We verify the **functional correctness** of self-timed circuits in terms of the relationship between their input and output sequences.

Our formalization considers a general case where:

- the initial state may contain some valid data; and
- there can be some valid data remaining in the final state.

Main theorem:

$$\mathit{extract}(\mathit{run}(\mathit{inputs-seq}, \mathit{st}, n)) \mathit{++} \mathit{out-seq} = \\ \mathit{op-map}(\mathit{in-seq}) \mathit{++} \mathit{extract}(\mathit{st})$$

Outline

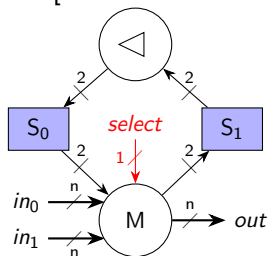
- 1 Progress
- 2 Verification Flow
- 3 Arbitrated Merge**
- 4 Conclusion

Arbitrated Merge

Arbitrated merge is a well-known self-timed circuit model that provides **mutually exclusive access** to a shared resource.

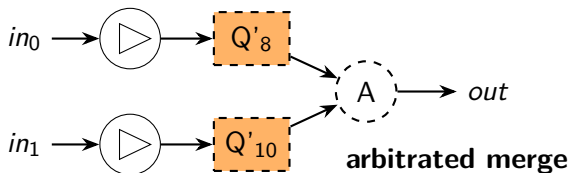
Produce **non-deterministic output sequences** due to arbitrary arrival times of requests.

We formalize an arbitrated merge joint that provides mutually exclusive access to its output link from its two input links on a **first-come-first-served** basis [Roncken et al.:2017].

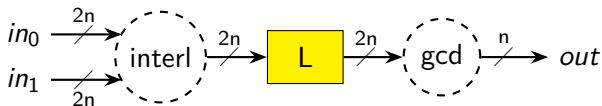


Circuits Performing Arbitrated Merges

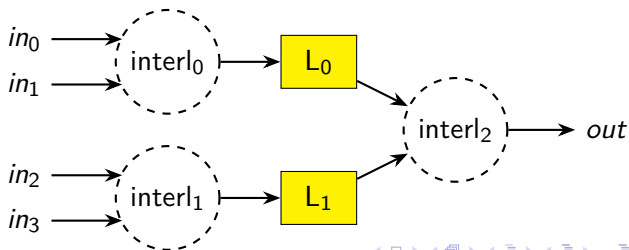
interl



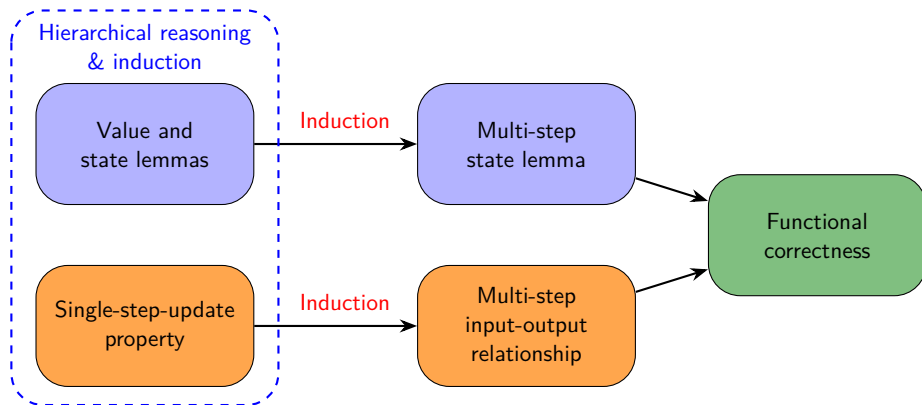
interl-gcd



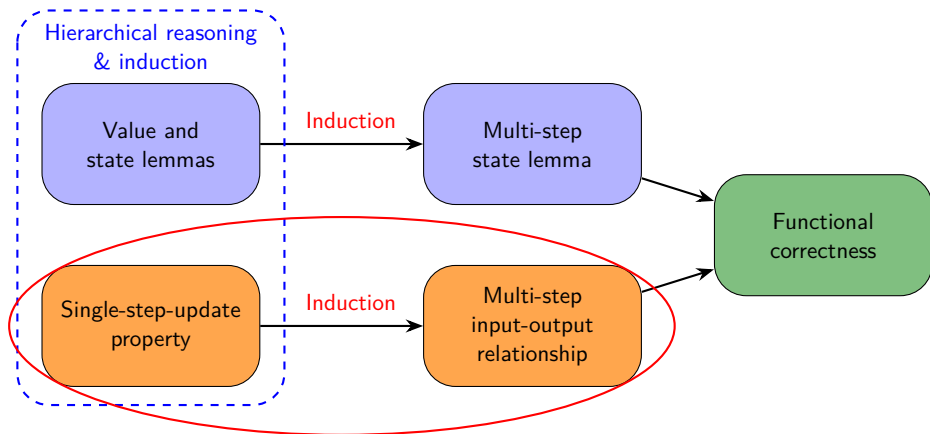
comp-interl



Verification Flow



Verification Flow



Arbitrated Merge Verification

Define **two extraction functions** for each arbitrated merge, one for each input stream.

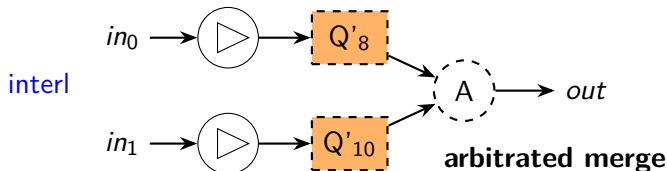
Single-step-update property:

$$\mathit{extract}_0(\mathit{step}(\mathit{inputs}, st)) = \mathit{extracted}_0\text{-step}(\mathit{inputs}, st)$$

$$\mathit{extract}_1(\mathit{step}(\mathit{inputs}, st)) = \mathit{extracted}_1\text{-step}(\mathit{inputs}, st)$$

Arbitrated Merge Verification

The multi-step input-output relationship is established using the **membership relation** (\in) and the **interleaving operation** (\otimes).



$interl\$extract_0$ and $interl\$extract_1$ extract valid data from two complex links Q'_8 and Q'_{10} , respectively.

let $st_f := interl\$run(inputs-seq, st, n)$,

$\forall x \in (interl\$extract_0(st_f) \otimes interl\$extract_1(st_f))$.

$$(x ++ out-seq) \in \left((in_0-seq ++ interl\$extract_0(st)) \otimes (in_1-seq ++ interl\$extract_1(st)) \right)$$

Outline

- 1 Progress
- 2 Verification Flow
- 3 Arbitrated Merge
- 4 Conclusion**




Conclusion

Reviewed the progress of our work on developing a hierarchical method for self-timed circuit modeling and verification.

Presented the main steps in our verification flow.

Discussed our strategy for verifying self-timed circuits performing arbitrated merge operations.

Developed a library that supports reasoning about the membership relation and the interleaving operation.

-  C. Chau, W. A. Hunt Jr., M. Kaufmann, M. Roncken, and I. Sutherland (2018)
Data-Loop-Free Self-Timed Circuit Verification
ASYNC 2018, 51 – 58.
-  C. Chau, W. A. Hunt Jr., M. Roncken, and I. Sutherland (2017)
A Framework for Asynchronous Circuit Modeling and Verification in ACL2
HVC 2017, 3 – 18.
-  M. Roncken, I. Sutherland, C. Chen, Y. Hei, W. Hunt Jr., and C. Chau, with S. M. Gilla, H. Park, X. Song, A. He, and H. Chen (2017)
How to Think about Self-Timed Systems
Asilomar 2017, 1597 – 1604.

Questions?