# A Review of the Self-Timed Circuit Verification Framework

**Cuong Chau**

*ckcuong@cs.utexas.edu*

Department of Computer Science
The University of Texas at Austin

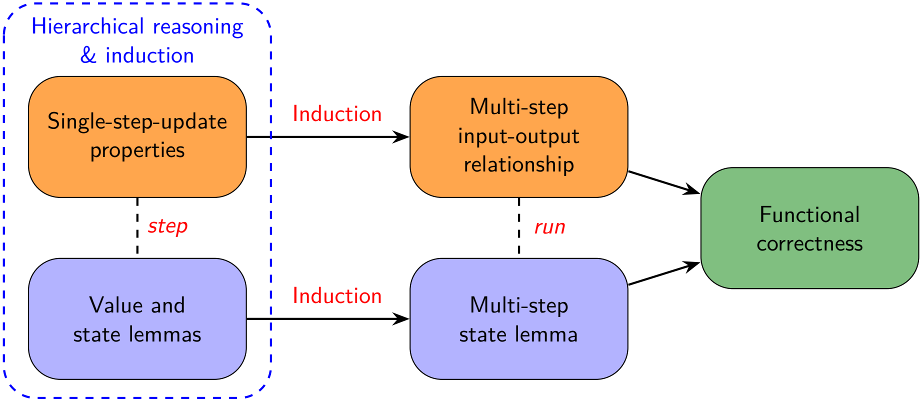
ACL2 Seminar Talk

November 30, 2018

## Outline

## Outline

# Verification Flow


Hierarchical reasoning & induction
Single-step-update properties
Induction
Multi-step input-output relationship
step
run
Value and state lemmas
Induction
Multi-step state lemma
Functional correctness

## Value and State Lemmas

**Value lemma:** characterize the module's outputs
$se(module\text{-}name, inputs, st, netlist) = outputs(inputs, st)$

**State lemma:** characterize the module's next state
$de(module\text{-}name, inputs, st, netlist) = step(inputs, st)$

Functions *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.

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

## Multi-Step State Lemma

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

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

```
$run(inputs\text{-}seq, st, n) :=$
  **if** $(n \leq 0)$ $st$
  **else**
    $run(rest(inputs\text{-}seq),$
        $step(first(inputs\text{-}seq), st),$
        $n - 1)$
```

## 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 **valid data residing in state** $st$.

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

Specify the input-output relationship after **one execution step**.

Introduce an extraction function for each self-timed module, $\mathit{extract}(\mathit{st})$, that returns a sequence of values computed from **valid data residing in state** *st*.

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

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

## Single-Step-Update Property

***Example:*** 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).

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

where

- $++$ is the concatenation operator;
- *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.

***Example:***

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

## 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.

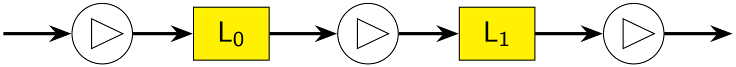
***Example:***

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

The functional correctness theorem is a direct corollary of the multi-step input-output relationship that is stated in terms of the *de-n* function, while that relationship is formalized in terms of the *run* function.

## Outline

## A FIFO Queue of Two Links



## Outline

## Arbitrated Merge

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

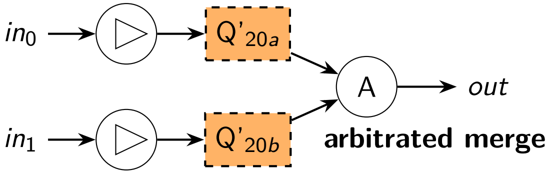
Produce non-deterministic output sequences due to arbitrary arrival times of requests.

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[1].

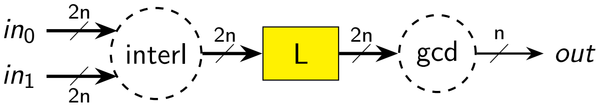
[1] M. Roncken et al. "How to Think about Self-Timed Systems". In: *Proc of the Fifty First IEEE Asilomar Conference on Signals, Systems, and Computers (Asilomar-2017)*. 2017, pp. 1597–1604.

# Circuits Performing Arbitrated Merges

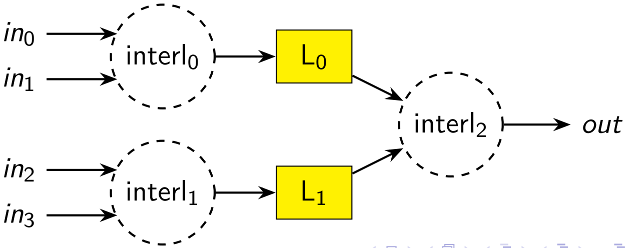
interl



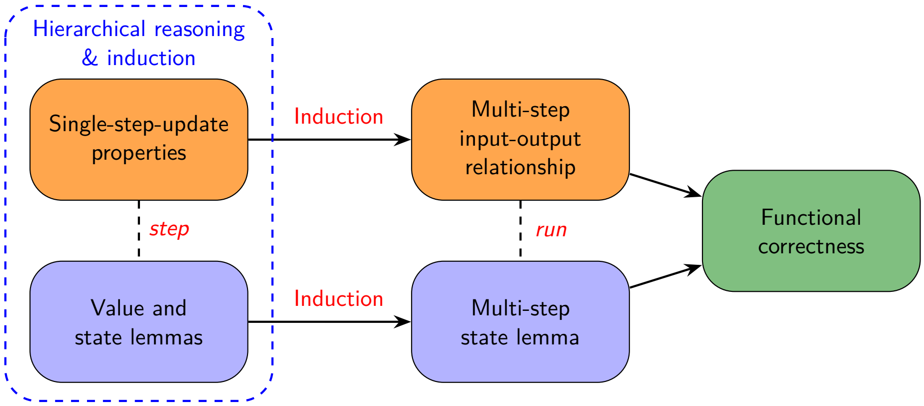
interl-gcd



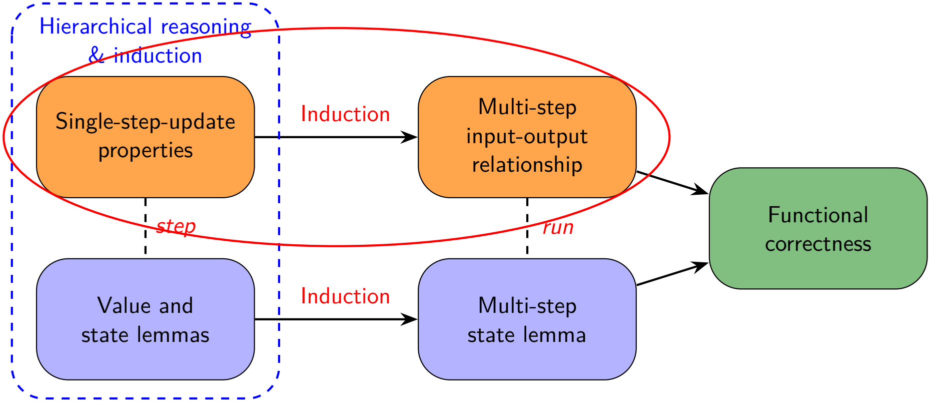
comp-interl



# Verification Flow


Hierarchical reasoning & induction
Single-step-update properties
step
Value and state lemmas
Induction
Induction
Multi-step input-output relationship
run
Multi-step state lemma
Functional correctness

# Verification Flow


Hierarchical reasoning & induction
Single-step-update properties
Induction
Multi-step input-output relationship
step
run
Value and state lemmas
Induction
Multi-step state lemma
Functional correctness

## Arbitrated Merge Verification

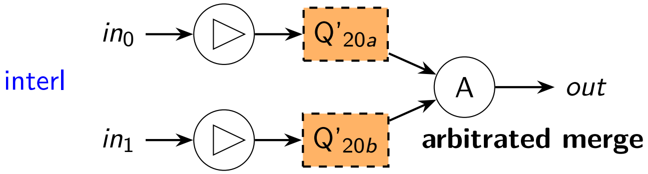
Define **two extraction functions** for each arbitrated merge, one for each input stream.

**Single-step-update properties:**

$$extract_0(step(inputs, st)) = extracted_0\text{-}step(inputs, st)$$
$$extract_1(step(inputs, st)) = extracted_1\text{-}step(inputs, st)$$

## 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 $Q'_{20a}$ and $Q'_{20b}$, respectively.

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

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

$$(x \mathbin{++} out\text{-}seq) \in \Big((in_0\text{-}seq \mathbin{++} interl\$extract_0(st)) \otimes (in_1\text{-}seq \mathbin{++} interl\$extract_1(st))\Big)$$

## Outline

## Conclusion

Reviewed our ACL2 verification framework for self-timed circuit designs.

Illustrated the framework through two examples.

- a self-timed circuit with **deterministic** outputs: a FIFO queue of two links; and
- a self-timed circuit with **non-deterministic** outputs: a circuit performing arbitrated merges.

Questions?