A Hierarchical Approach to Self-Timed Circuit Verification

Cuong Chau[1], Warren A. Hunt Jr.[1], Matt Kaufmann[1], Marly Roncken[2], and Ivan Sutherland[2]

{*ckcuong,hunt,kaufmann*}*@cs.utexas.edu,*

mroncken@pdx.edu, ivans@cecs.pdx.edu

[1] The University of Texas at Austin

[2] Portland State University

May 14, 2019

Motivation and Goals

Motivation:

- Many efforts in verifying self-timed circuit implementations concern **circuit-level timing properties** or **communication properties**.
- Most verification methods for self-timed circuits have concentrated on **small-size** circuits.
- Scalable methods for self-timed system verification are highly desirable.

Motivation and Goals

Motivation:

- Many efforts in verifying self-timed circuit implementations concern **circuit-level timing properties** or **communication properties**.
- Most verification methods for self-timed circuits have concentrated on **small-size** circuits.
- Scalable methods for self-timed system verification are highly desirable.

Goals:

- Develop scalable methods for reasoning about the functional correctness of self-timed circuits and systems, while **abstracting away circuit-level timing constraints**.
- Implement those methods using the ACL2 theorem proving system, providing a useful automated framework with associated libraries to support the mechanical analysis of general-purpose, self-timed circuit designs.

Approach

Extend the DE-based, synchronous-style verification system[1] to one that is capable of analyzing self-timed system models.

[1] W. A. Hunt Jr. "The DE Language". In: *Computer-Aided Reasoning: ACL2 Case Studies*. Springer US, 2000. Chap. 10, pp. 151–166.

[2] M. Roncken et al. "Naturalized Communication and Testing". In: *ASYNC-2015*, pp. 77–84.

Approach

Extend the DE-based, synchronous-style verification system[1] to one that is capable of analyzing self-timed system models.

Apply the link-joint model[2] to modeling self-timed circuit designs.

[1] W. A. Hunt Jr. "The DE Language". In: *Computer-Aided Reasoning: ACL2 Case Studies*. Springer US, 2000. Chap. 10, pp. 151–166.

[2] M. Roncken et al. "Naturalized Communication and Testing". In: *ASYNC-2015*, pp. 77–84.

Approach

Extend the DE-based, synchronous-style verification system[1] to one that is capable of analyzing self-timed system models.

Apply the link-joint model[2] to modeling self-timed circuit designs.

Develop a hierarchical (compositional) reasoning approach that is amenable to verifying correctness of **large**, **non-deterministic** systems without a large growth of the time complexity.

[1] W. A. Hunt Jr. "The DE Language". In: *Computer-Aided Reasoning: ACL2 Case Studies*. Springer US, 2000. Chap. 10, pp. 151–166.

[2] M. Roncken et al. "Naturalized Communication and Testing". In: *ASYNC-2015*, pp. 77–84.

Approach

Extend the DE-based, synchronous-style verification system[1] to one that is capable of analyzing self-timed system models.

Apply the link-joint model[2] to modeling self-timed circuit designs.

Develop a hierarchical (compositional) reasoning approach that is amenable to verifying correctness of **large**, **non-deterministic** systems without a large growth of the time complexity.

- Avoid exploring the operations **internal to a verified submodule** as well as their interleavings.
- The **input-output relationship** of a verified submodule is determined based on the communication signals at the submodule's input and output ports, while **abstracting away all execution paths internal to that submodule**.

[1] W. A. Hunt Jr. "The DE Language". In: *Computer-Aided Reasoning: ACL2 Case Studies*. Springer US, 2000. Chap. 10, pp. 151–166.

[2] M. Roncken et al. "Naturalized Communication and Testing". In: *ASYNC-2015*, pp. 77–84.

Contributions

Extend our previous framework[3] to model and verify circuit generators with **parameterized data sizes**.

Demonstrate that our verification framework is applicable to circuits with **loops** as well.

Formalize an (non-deterministically) arbitrated merge joint that provides **mutually exclusive access** to its output link from its two input links.

Develop strategies for verifying the functional correctness of self-timed circuits performing arbitrated merges.

[3] C. Chau et al. "Data-Loop-Free Self-Timed Circuit Verification". In: *ASYNC-2018*, pp. 51–58.

Outline

Outline

DE System

DE is a formal occurrence-oriented hardware description language developed in ACL2 for describing **finite-state machines**.

DE System

DE is a formal occurrence-oriented hardware description language developed in ACL2 for describing **finite-state machines**.

The semantics of the DE language is given by a simulator that computes the **outputs** and **next state** for a module from the module's **current inputs** and **current state**.

DE System

DE is a formal occurrence-oriented hardware description language developed in ACL2 for describing **finite-state machines**.

The semantics of the DE language is given by a simulator that computes the **outputs** and **next state** for a module from the module's **current inputs** and **current state**.

In our self-timed modeling approach, we invoke the DE simulator whenever any primary input changes.

> Allow the design to proceed at a rate moderated by oracle values — extra input values modeling **non-determinacy** — that can cause any part of the logic to **delay an arbitrary amount**.

DE System

DE is a formal occurrence-oriented hardware description language developed in ACL2 for describing **finite-state machines**.

The semantics of the DE language is given by a simulator that computes the **outputs** and **next state** for a module from the module's **current inputs** and **current state**.

In our self-timed modeling approach, we invoke the DE simulator whenever any primary input changes.

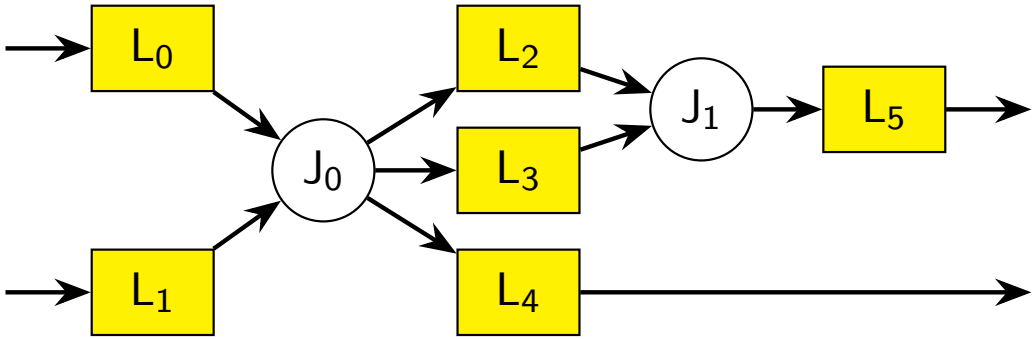
> Allow the design to proceed at a rate moderated by oracle values — extra input values modeling **non-determinacy** — that can cause any part of the logic to **delay an arbitrary amount**.

We extended the DE primitive database with a new primitive that models the **validity of data** stored in a communication link.

Outline

Link-Joint Model

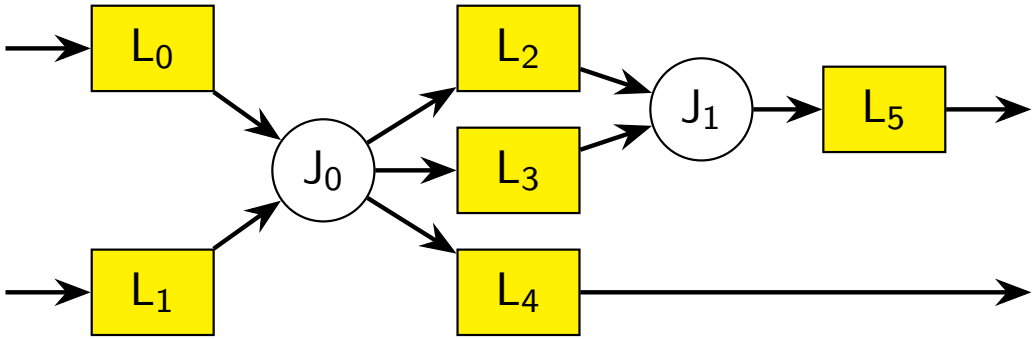
We model self-timed systems as **finite-state machines** representing networks of communication links and computation joints.



Links communicate with each other locally via joints using the **link-joint model**.

Link-Joint Model

We model self-timed systems as **finite-state machines** representing networks of communication links and computation joints.

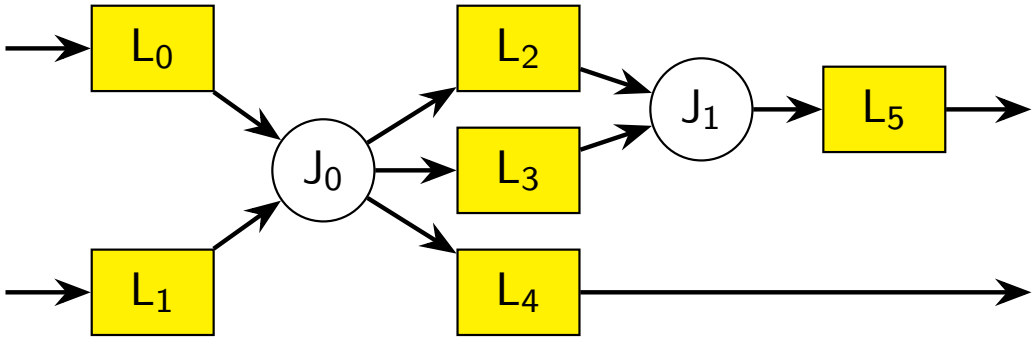


Links communicate with each other locally via joints using the **link-joint model**.

- Links are communication channels in which **data** are stored along with **a full/empty signal**.
- Joints implement **data operations** and **flow control**.
- A link connects exactly to one input and one output joint.

Link-Joint Model

We model self-timed systems as **finite-state machines** representing networks of communication links and computation joints.

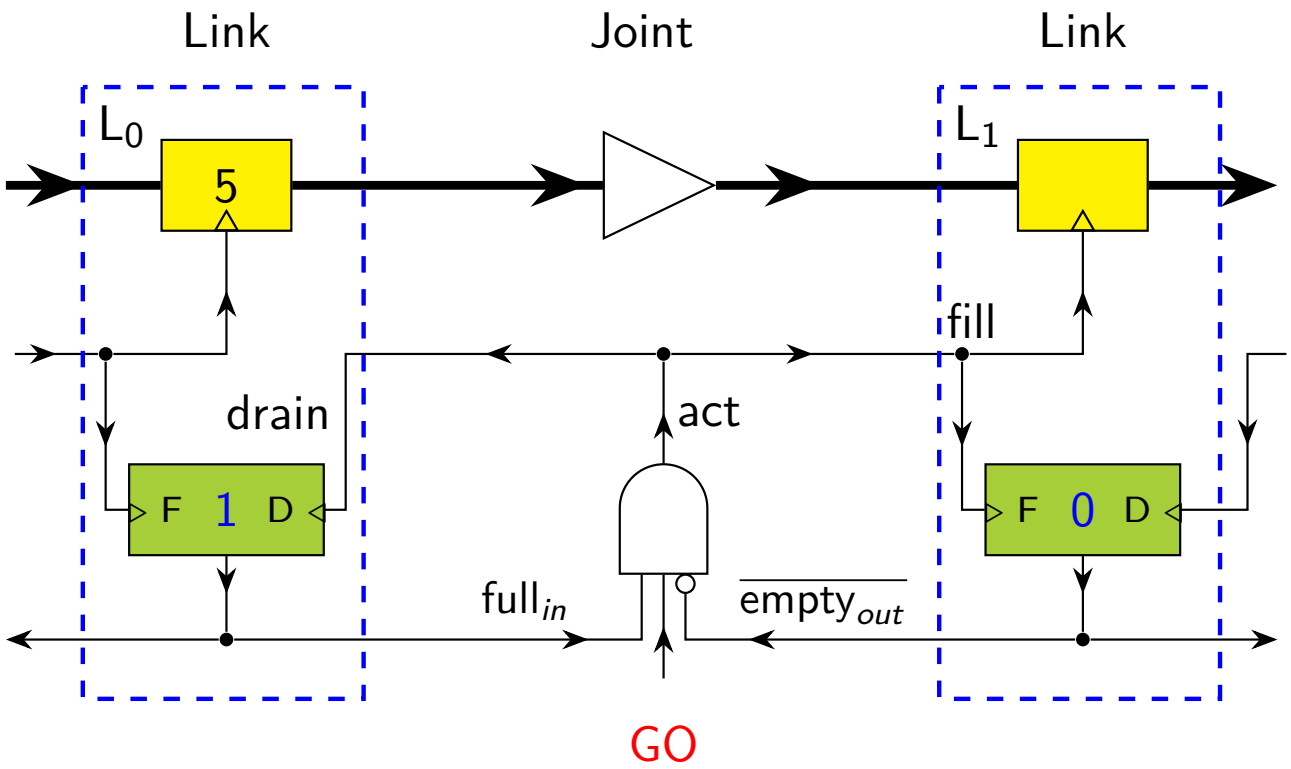


Links communicate with each other locally via joints using the **link-joint model**.

- Links are communication channels in which **data** are stored along with **a full/empty signal**.
- Joints implement **data operations** and **flow control**.
- A link connects exactly to one input and one output joint.

Necessary conditions for a **joint-action** to fire: all input and output links of that action are **full** and **empty**, respectively.

Details of the Link-Joint Model

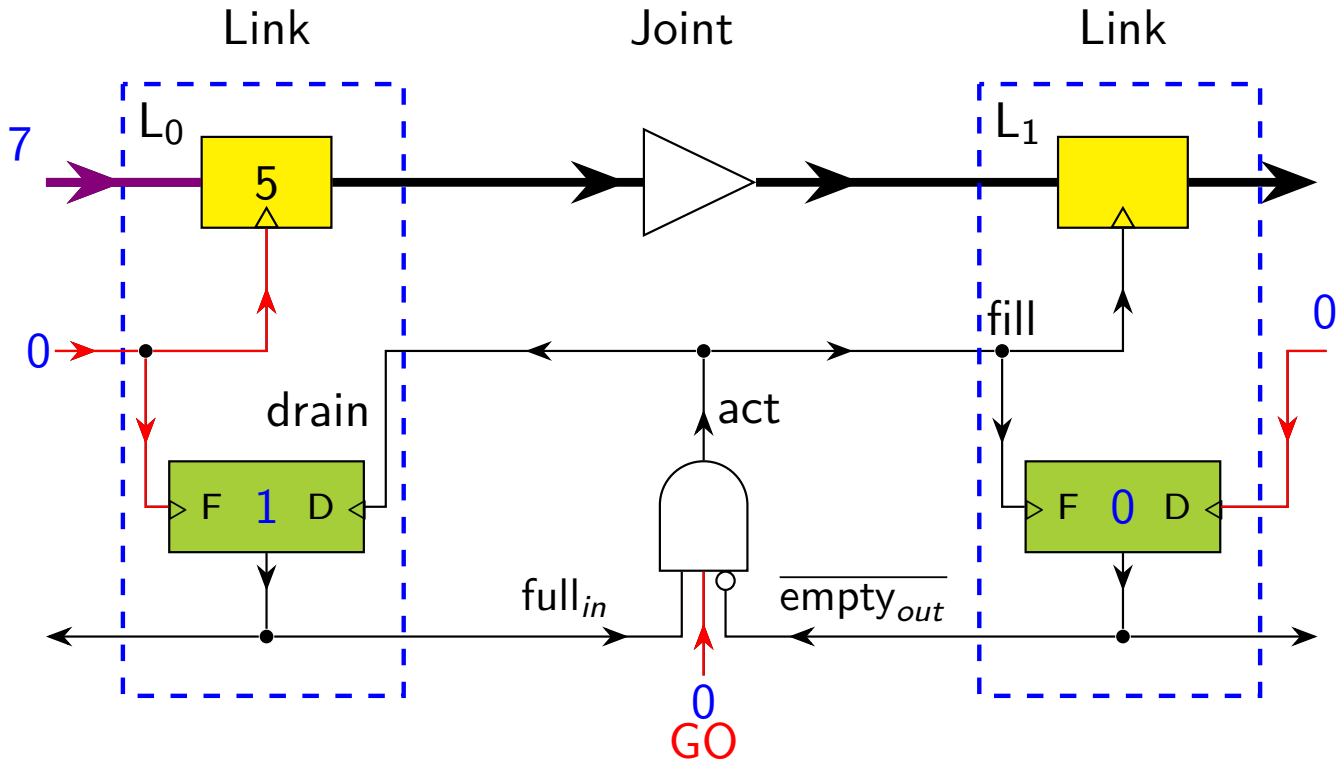


The green boxes represent instances of our new **DE** link-control primitive.

When a joint acts, three tasks will be executed in parallel:

- transfer data computed from the input links to the output links;
- fill (possibly a subset of) the output links, leaving them **full**;
- drain (possibly a subset of) the input links, leaving them **empty**.

Details of the Link-Joint Model (GO = 0)

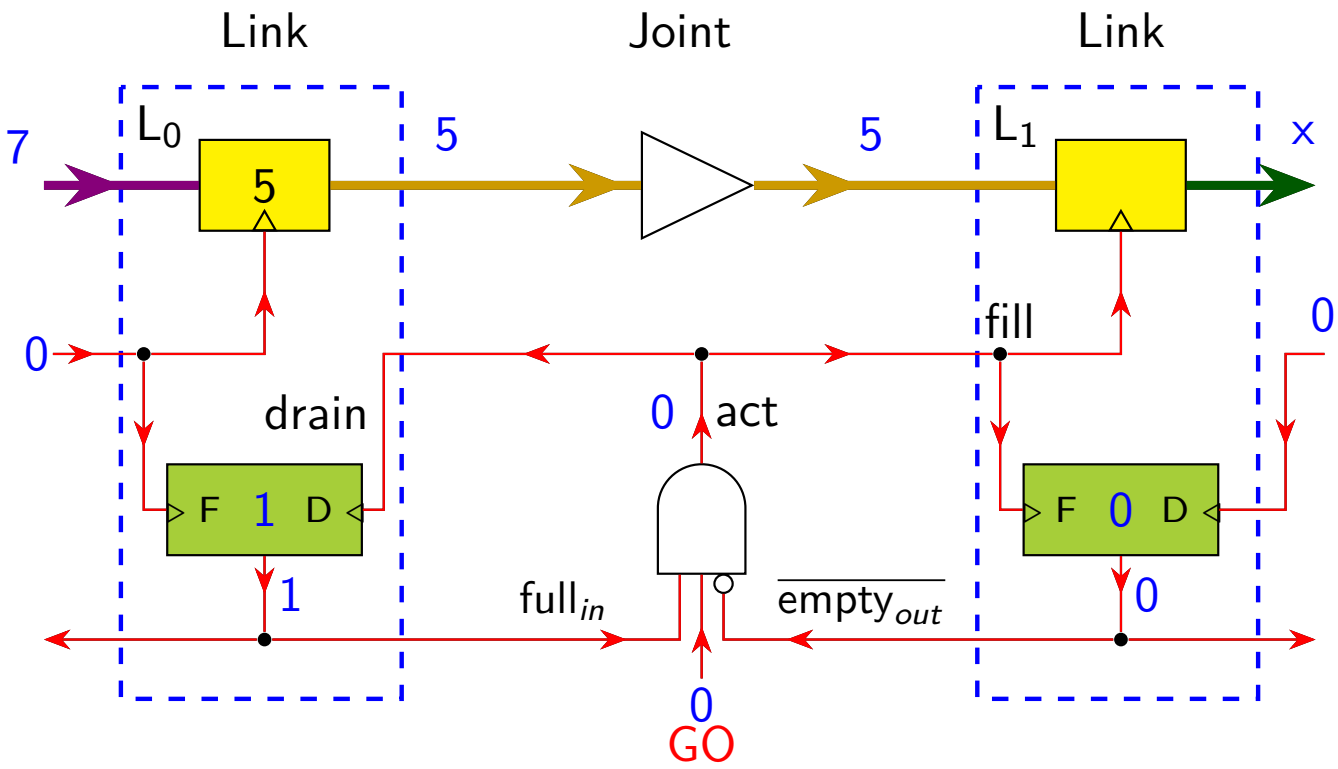


The green boxes represent instances of our new **DE** link-control primitive.

When a joint acts, three tasks will be executed in parallel:

- transfer data computed from the input links to the output links;
- fill (possibly a subset of) the output links, leaving them **full**;
- drain (possibly a subset of) the input links, leaving them **empty**.

Details of the Link-Joint Model (GO = 0)

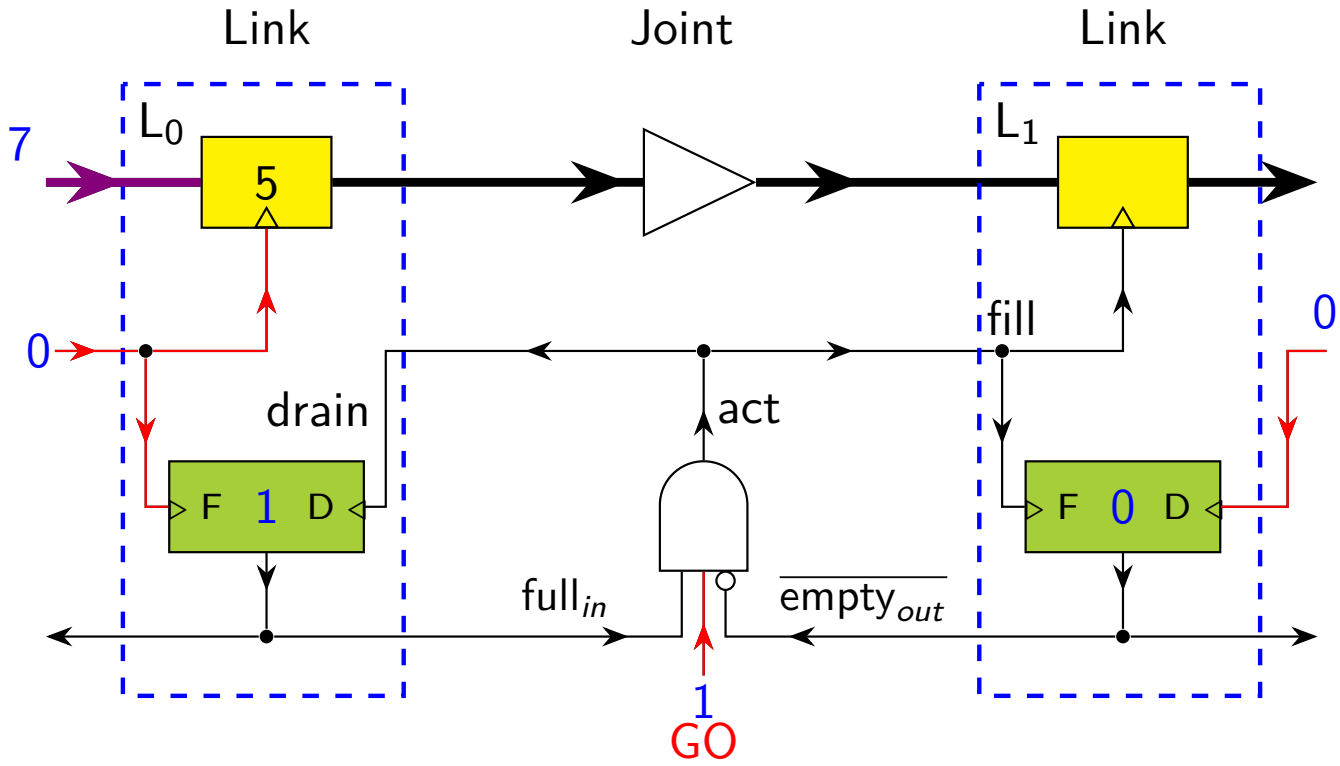


The green boxes represent instances of our new **DE** link-control primitive.

When a joint acts, three tasks will be executed in parallel:

- transfer data computed from the input links to the output links;
- fill (possibly a subset of) the output links, leaving them **full**;
- drain (possibly a subset of) the input links, leaving them **empty**.

Details of the Link-Joint Model (GO = 1)

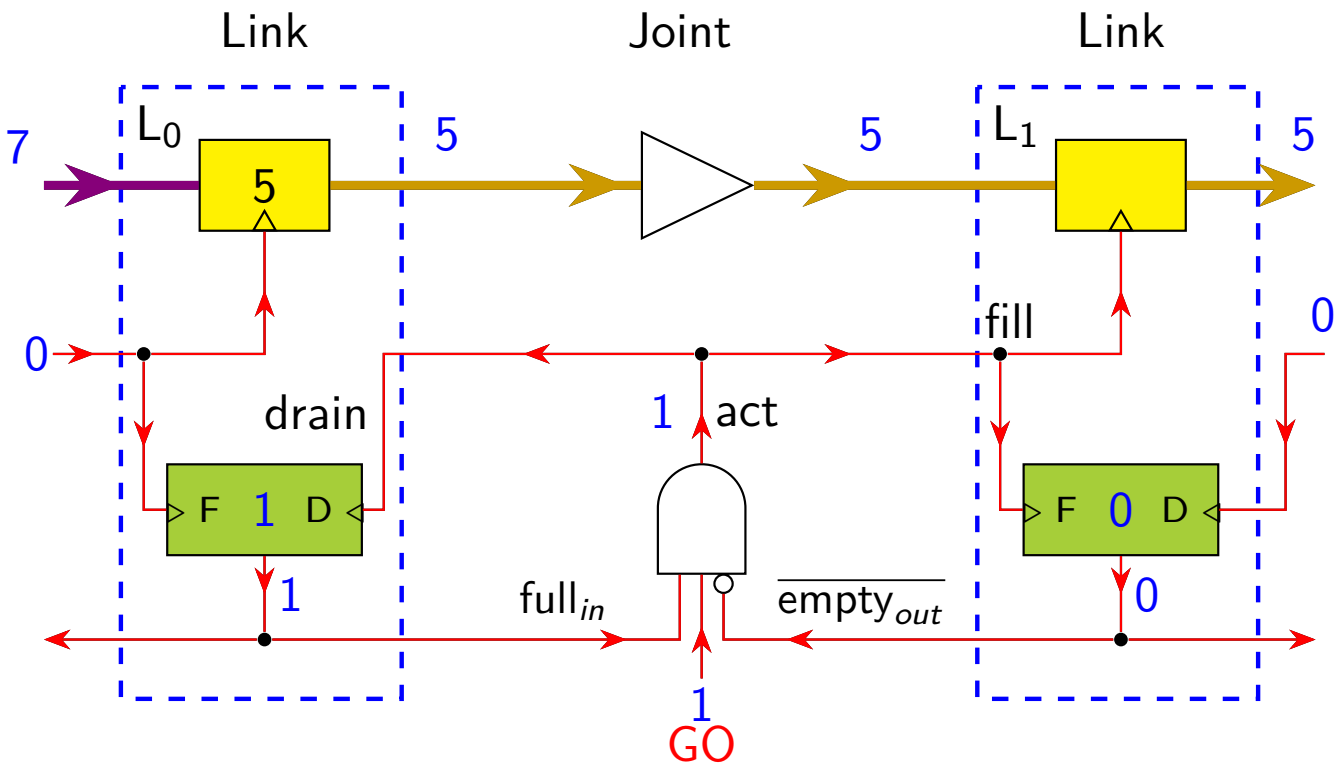


The green boxes represent instances of our new **DE** link-control primitive.

When a joint acts, three tasks will be executed in parallel:

- transfer data computed from the input links to the output links;
- fill (possibly a subset of) the output links, leaving them **full**;
- drain (possibly a subset of) the input links, leaving them **empty**.

Details of the Link-Joint Model (GO = 1)

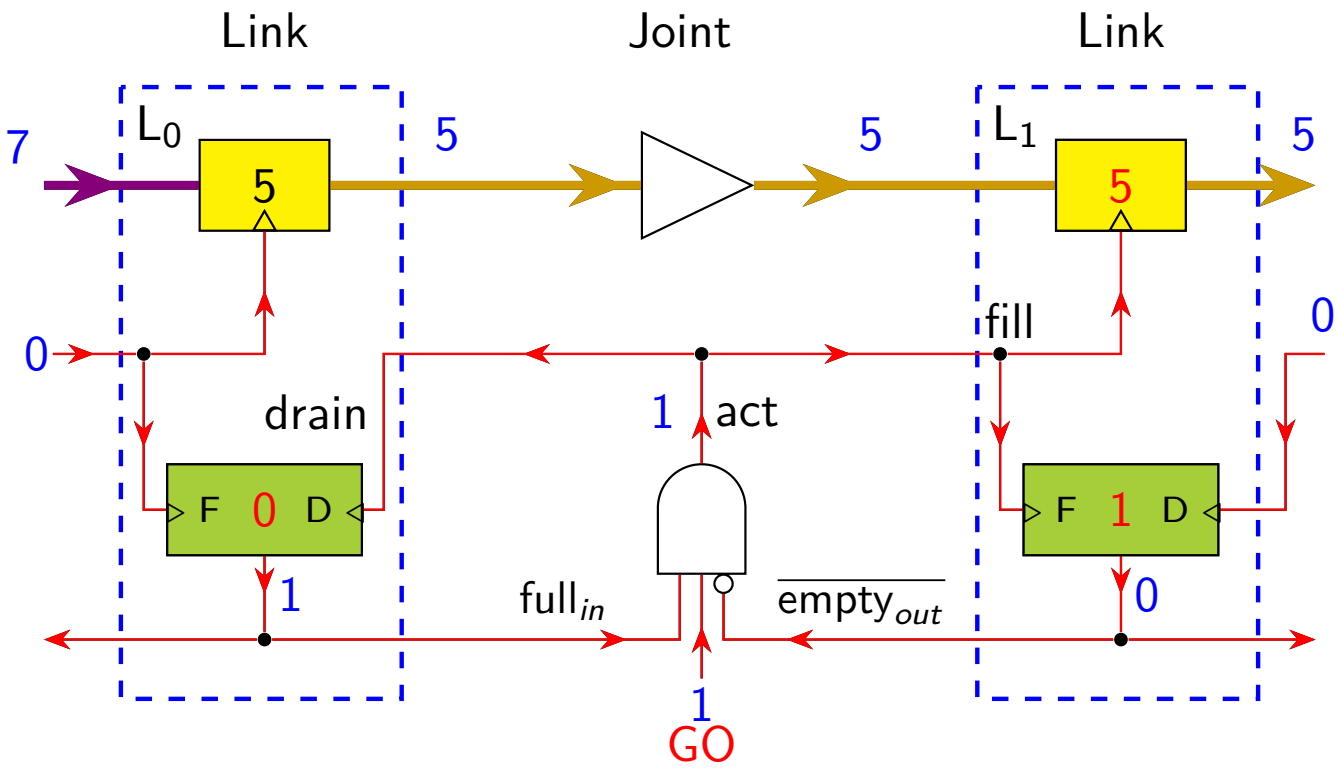


The green boxes represent instances of our new **DE** link-control primitive.

When a joint acts, three tasks will be executed in parallel:

- transfer data computed from the input links to the output links;
- fill (possibly a subset of) the output links, leaving them **full**;
- drain (possibly a subset of) the input links, leaving them **empty**.

Details of the Link-Joint Model (GO = 1)

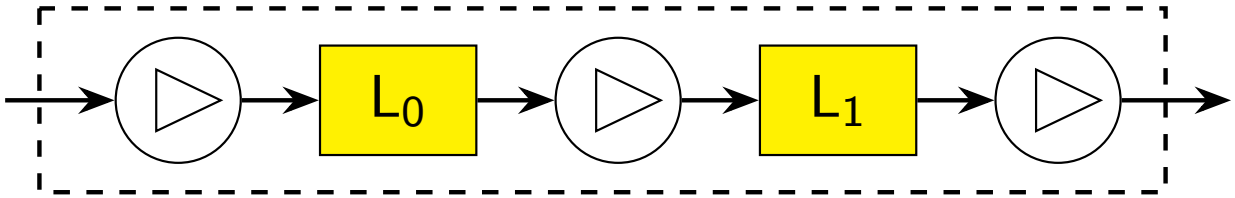


The green boxes represent instances of our new **DE** link-control primitive.

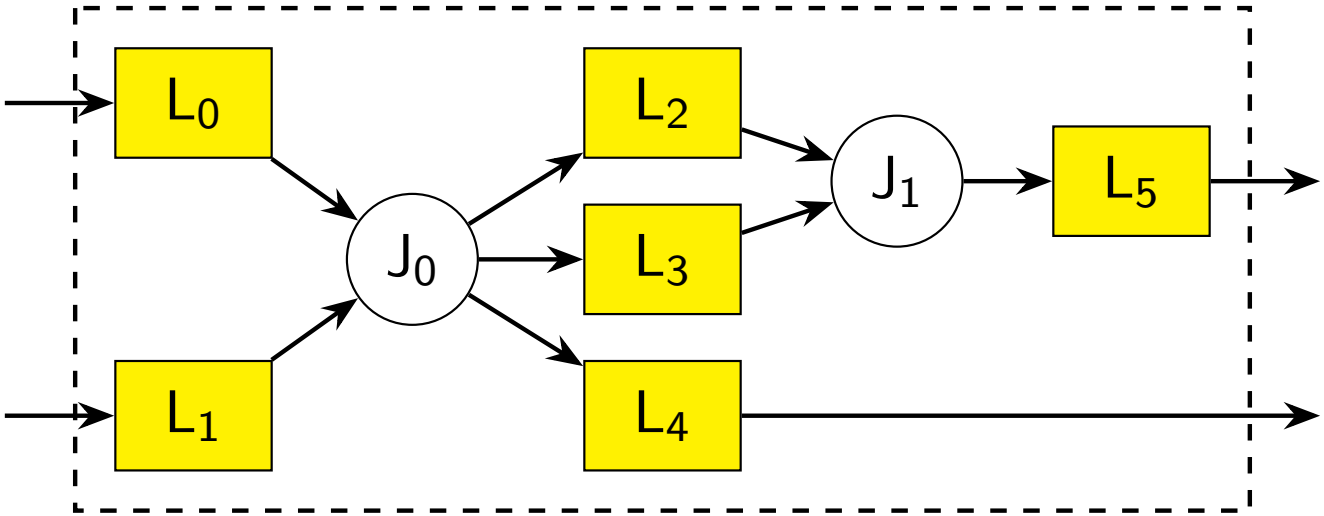
When a joint acts, three tasks will be executed in parallel:

- transfer data computed from the input links to the output links;
- fill (possibly a subset of) the output links, leaving them **full**;
- drain (possibly a subset of) the input links, leaving them **empty**.

Self-Timed Modules

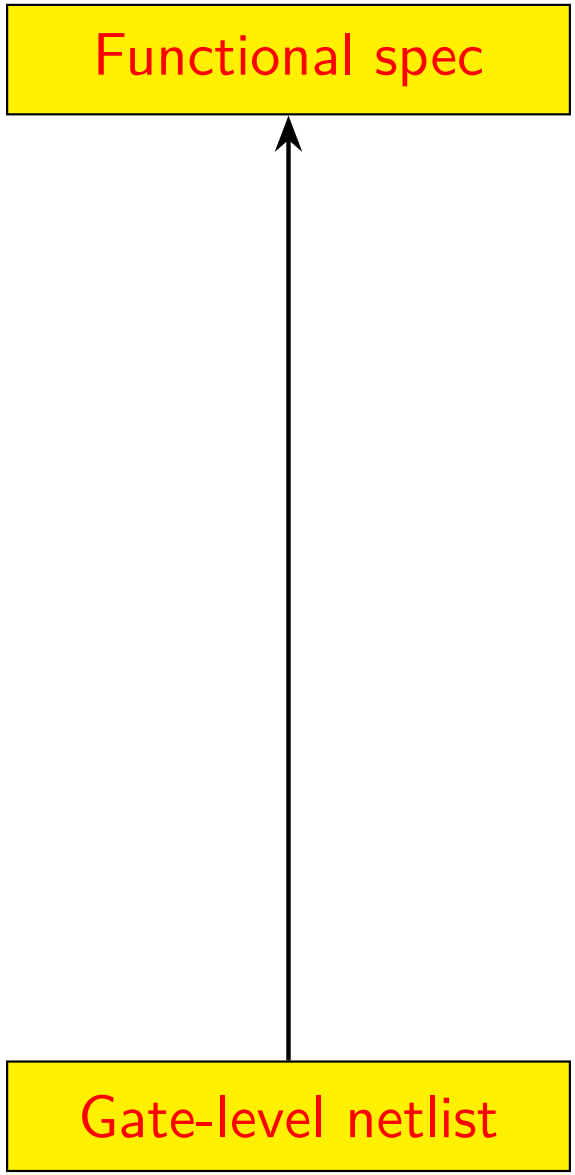


Complex joint: a queue of length two, $Q2$

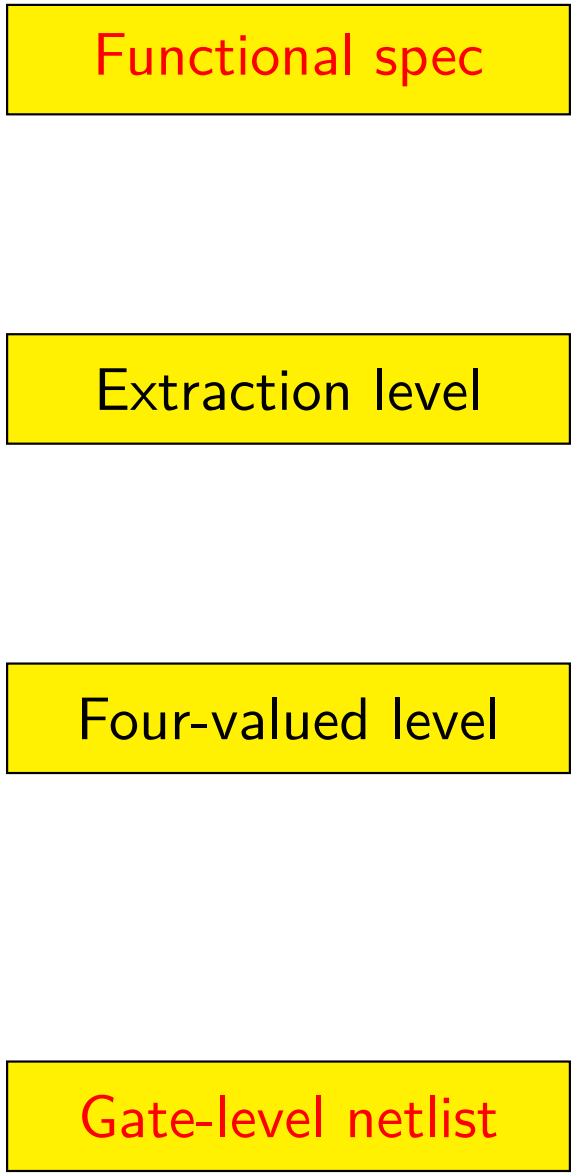


Complex link

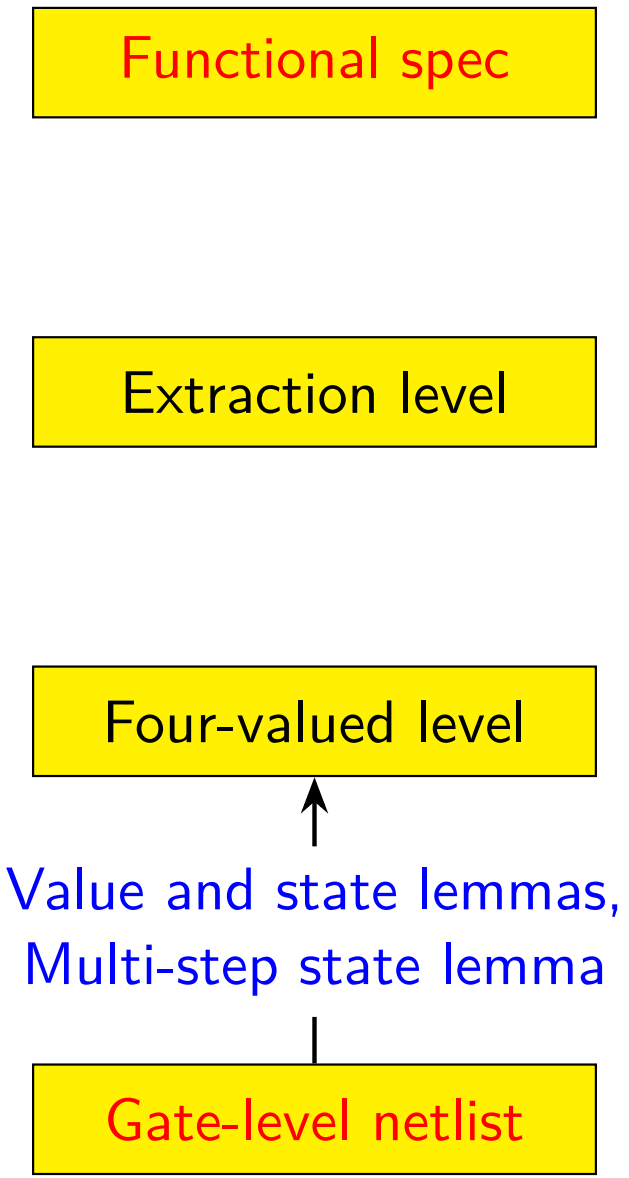
Verification Flow



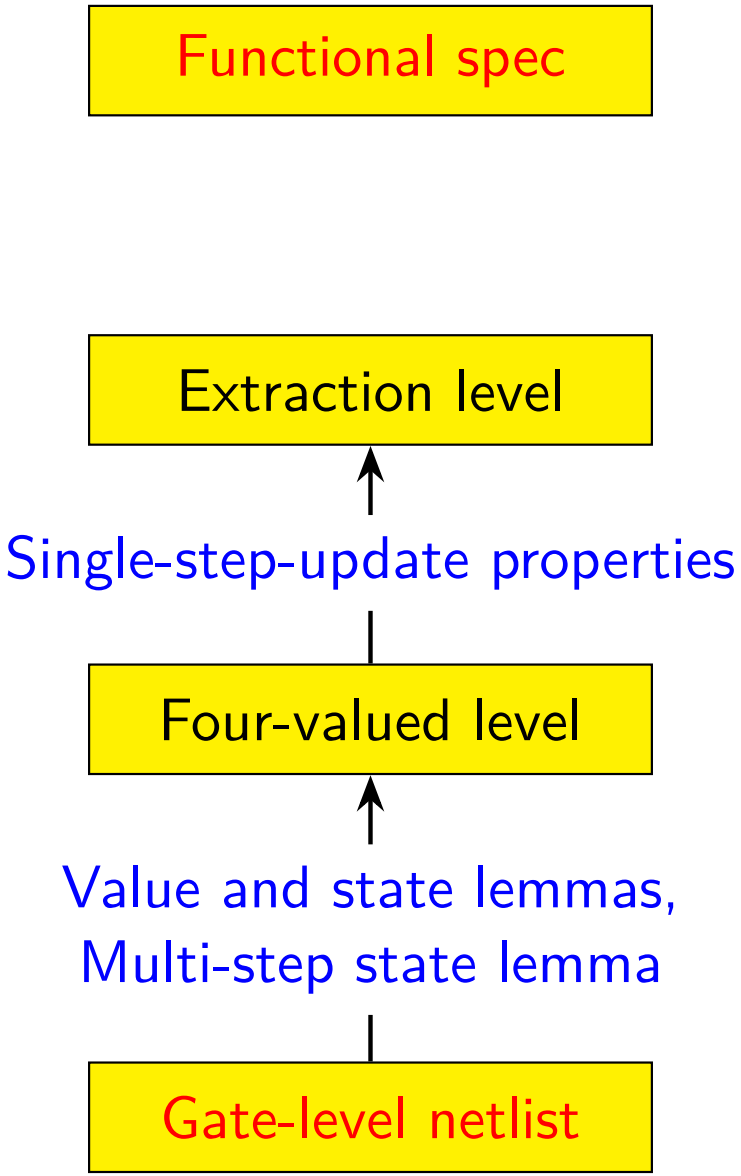
Verification Flow



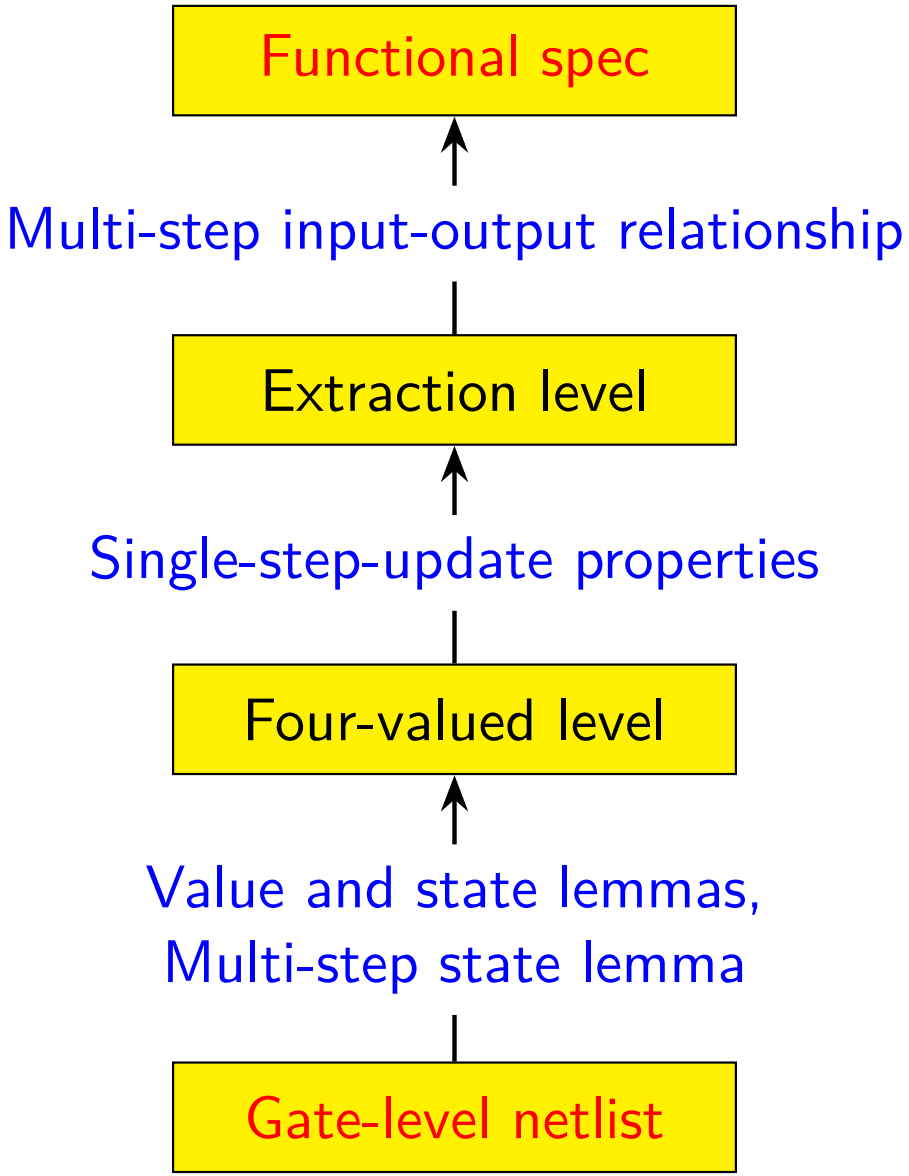
Verification Flow



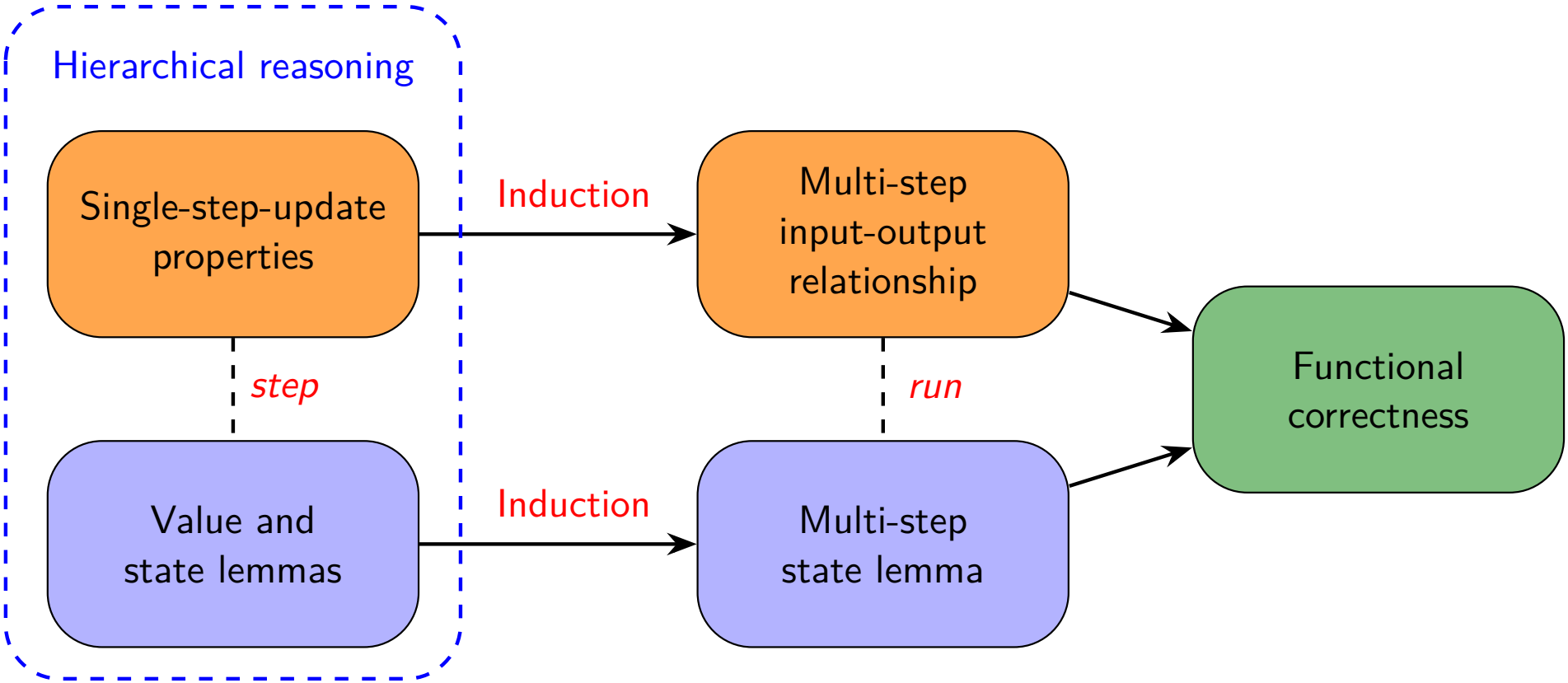
Verification Flow



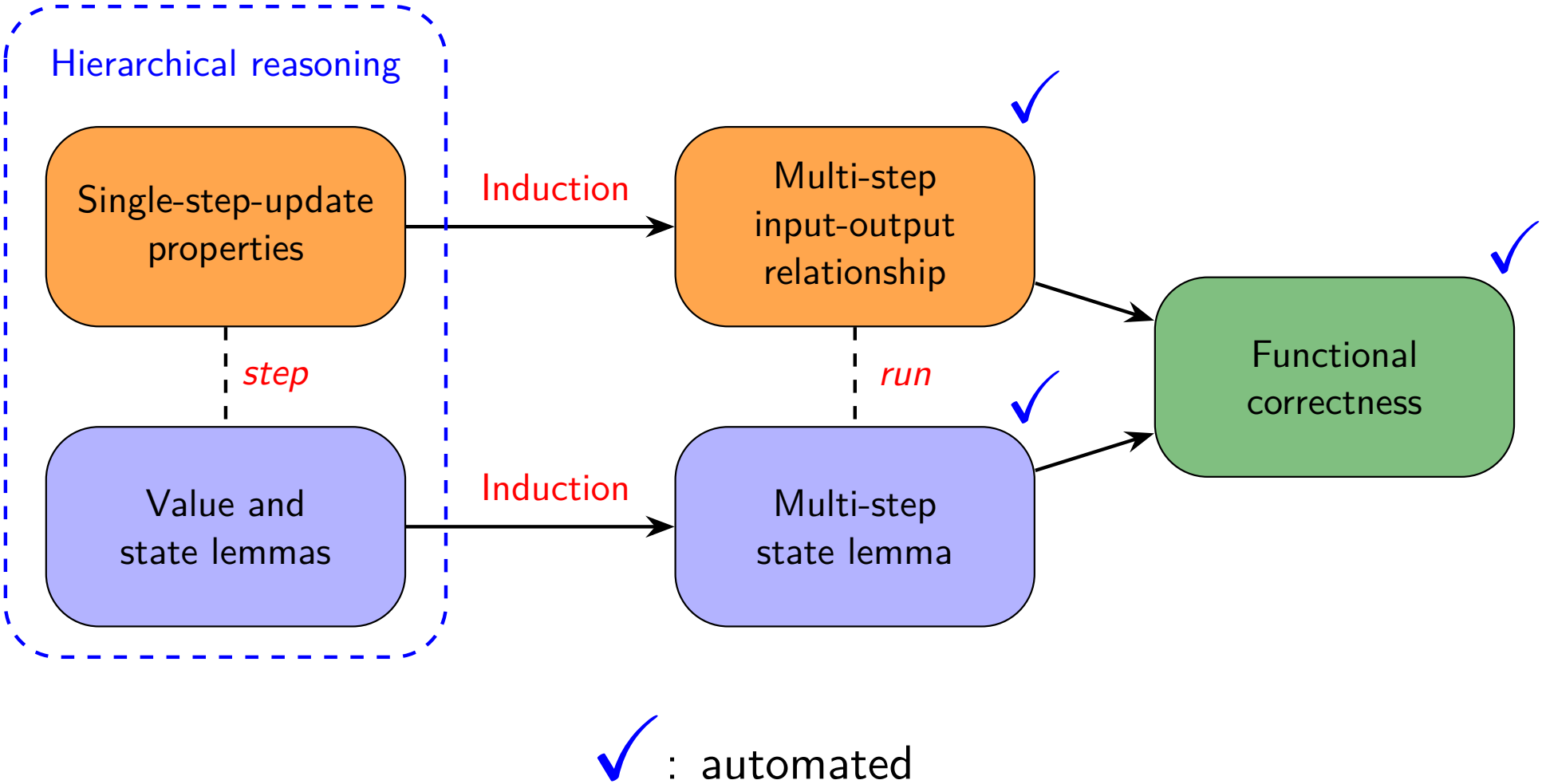
Verification Flow



Verification Steps

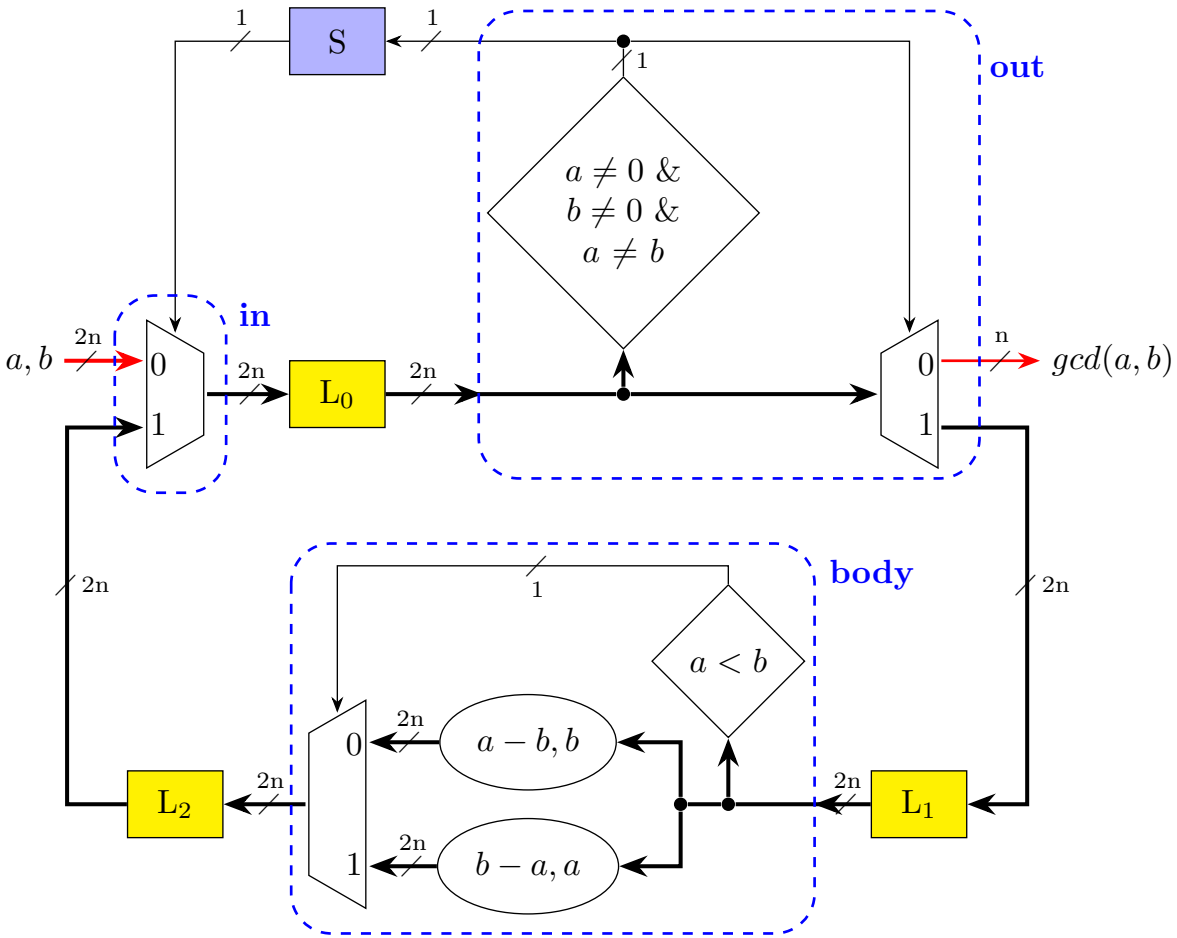


Verification Steps



Outline

A Greatest-Common-Divisor (GCD) Circuit Model



gcd-alg(*a*, *b*) :=

if $(a = 0)$ **then** *b*

else if $(b = 0)$ **then** *a*

else if $(a = b)$ **then** *a*

else if $(a < b)$ **then** *gcd-alg*$(b - a, a)$

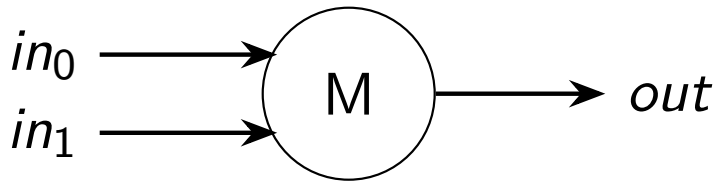
else *gcd-alg*$(a - b, b)$

Arbitrated Merge

Arbitrated merge, or arbiter, 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.

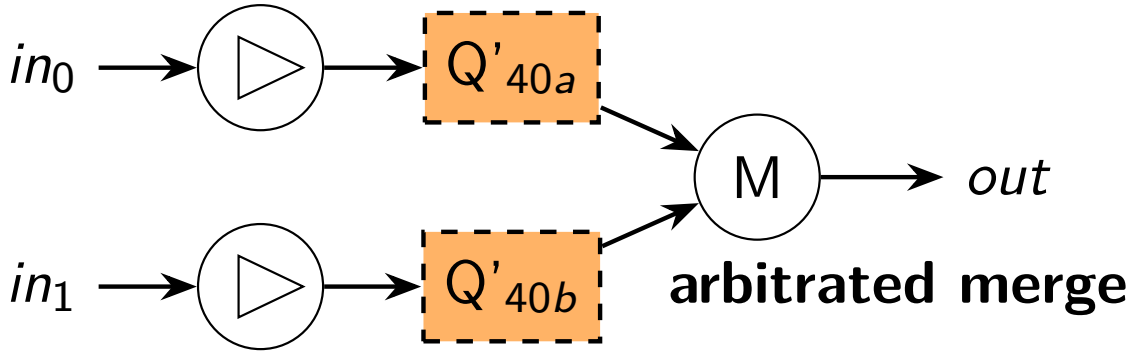
We formalize an arbitrated merge joint that provides mutually exclusive access to its output link from its two input links on a **first-come-first-served** basis[4].



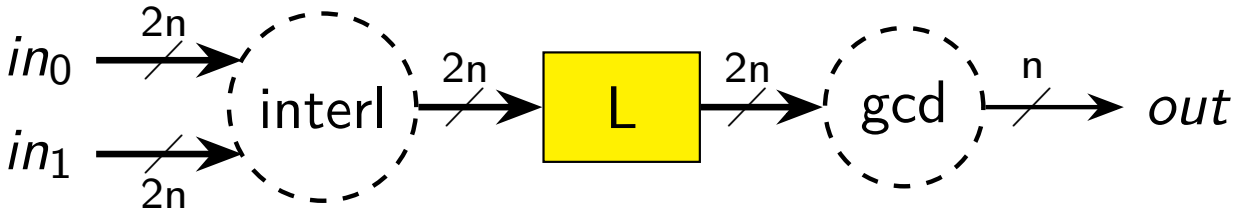
[4] M. Roncken et al. "How to Think about Self-Timed Systems". In: *Asilomar-2017*, pp. 1597–1604.

Circuits Performing Arbitrated Merges

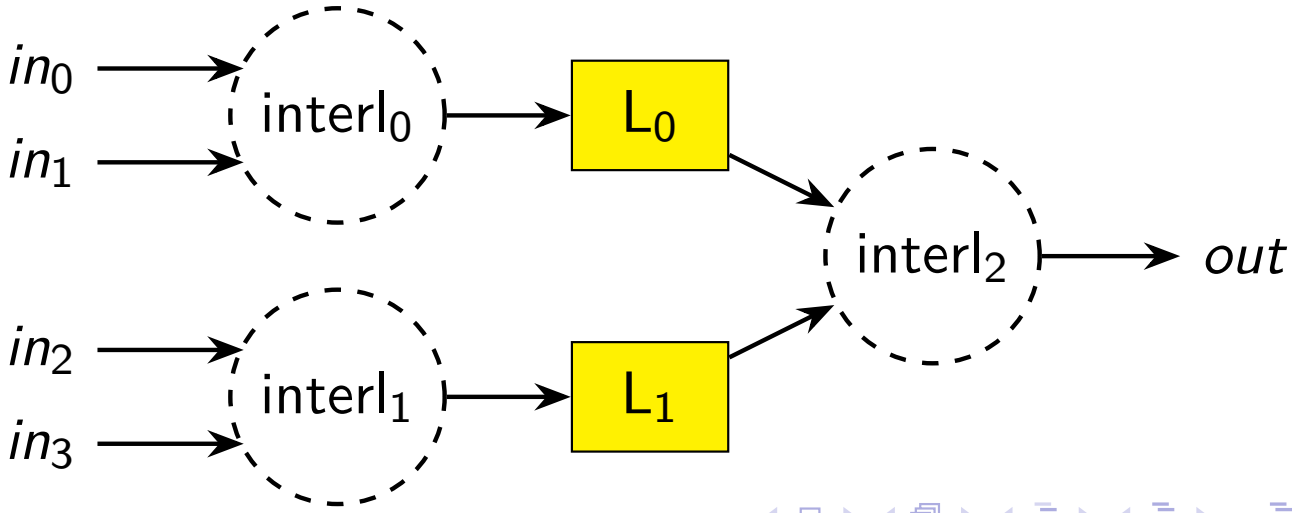
interl



igcd



comp-interl



Performance

Circuit	Proof time	# *go* signals	# *go* signals affecting reasoning
gcd	8s	3	3
Q5'	8s	4	4
Q10'	3s	9	1
Q20'	3s	19	1
Q40'	3s	39	1
interl	5s	81	3
igcd	12s	84	5
comp-interl	23s	243	9

Outline

Future Work

Implement a syntactic checker that detects link-joint topology violations in self-timed circuit designs.

Enhance the effectiveness of our framework by increasing automation through the further introduction of macros.

- Automate the proofs of value and state lemmas.

Apply our methodology to modeling self-timed microprocessors and verifying their functional properties.

- E.g., model and verify a self-timed version of the **FM9001** microprocessor.

Develop methods for analyzing mixed self-timed, synchronous circuits and systems.

Conclusions

We have developed a hierarchical, mechanized methodology that is capable of verifying the **functional correctness** of self-timed circuit designs at scale.

We model self-timed systems as networks of links communicating with each other locally via joints, using the link-joint model.

We model the **non-determinism of event-ordering** in self-timed circuits by associating each joint action with an external go signal that, when disabled, prevents that action from **firing**.

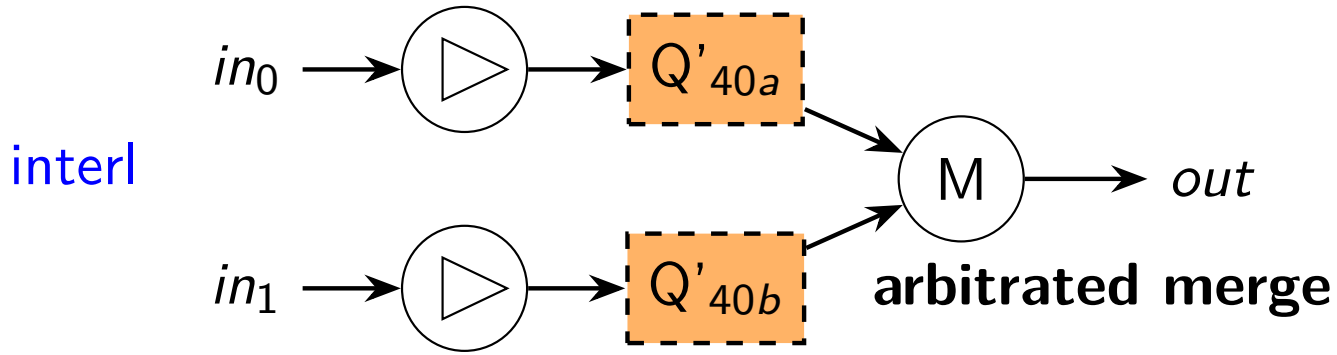
Successfully applied our modeling and verification approach to a sequence of increasingly complex self-timed circuit models.

- Data-loop-free circuits
- Iterative circuits
- Circuits involving non-deterministically arbitrated merges

Questions?

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'_{40a} and Q'_{40b}, respectively.

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

$\forall x \in (\mathit{interl\$extract}_0(st_f) \otimes \mathit{interl\$extract}_1(st_f))$.

$$(x +\!\!+ \mathit{out\text{-}seq}) \in \Big((\mathit{in}_0\text{-}\mathit{seq} +\!\!+ \mathit{interl\$extract}_0(st)) \otimes (\mathit{in}_1\text{-}\mathit{seq} +\!\!+ \mathit{interl\$extract}_1(st))\Big)$$