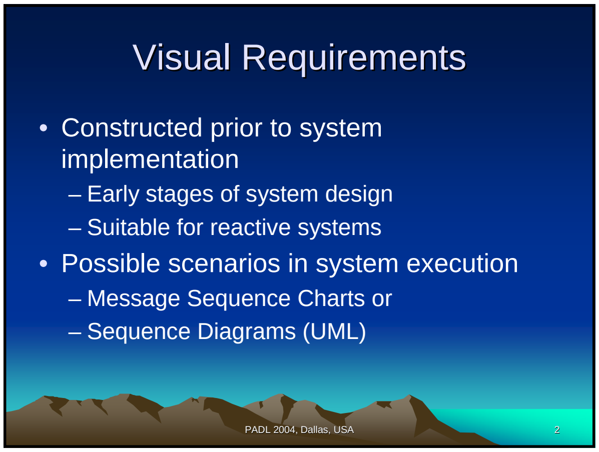
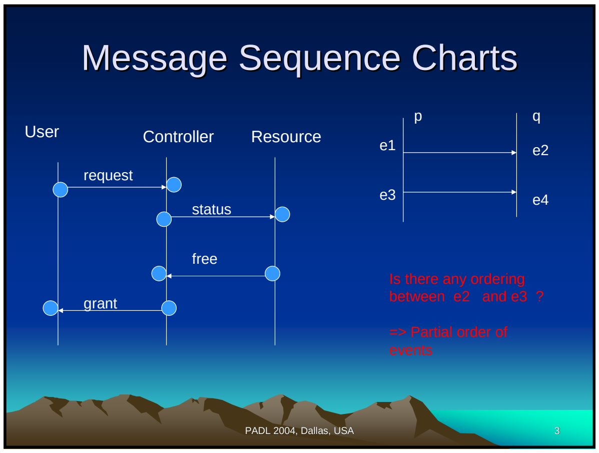
Symbolic Execution of Behavioral Requirements

Tao Wang, Abhik Roychoudhury,
Roland Yap, S.C. Choudhary
National University of Singapore



Visual Requirements

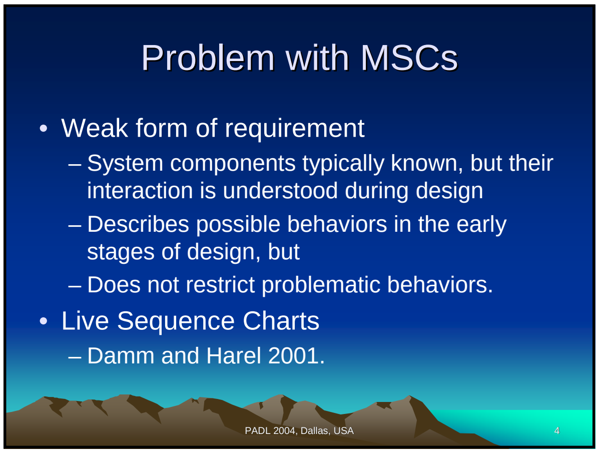
- Constructed prior to system implementation
  - Early stages of system design
  - Suitable for reactive systems
- Possible scenarios in system execution
  - Message Sequence Charts or
  - Sequence Diagrams (UML)



Message Sequence Charts

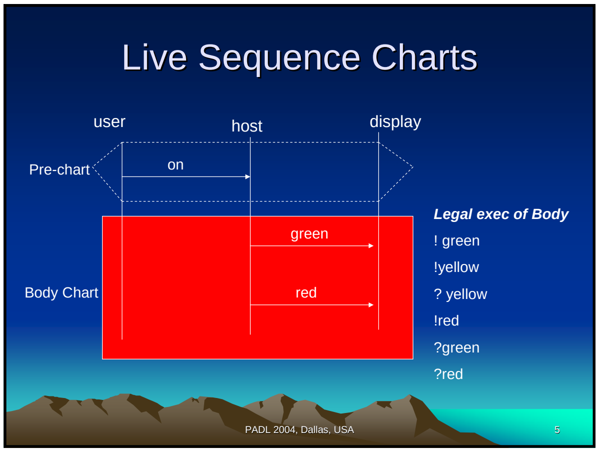
Is there any ordering between e2 and e3 ?

=> Partial order of events



Problem with MSCs

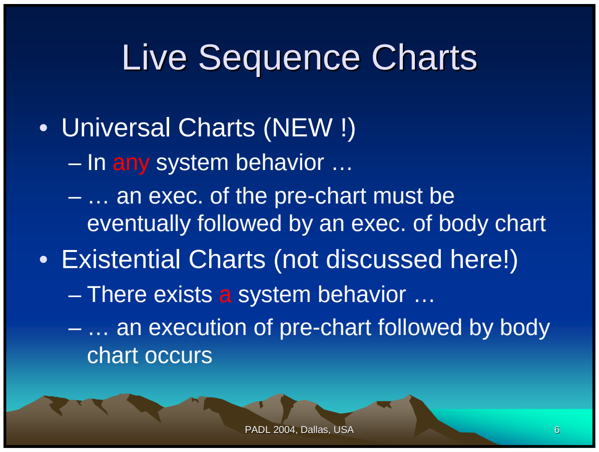
- Weak form of requirement
  - System components typically known, but their interaction is understood during design
  - Describes possible behaviors in the early stages of design, but
  - Does not restrict problematic behaviors.
- Live Sequence Charts
  - Damm and Harel 2001.



Live Sequence Charts

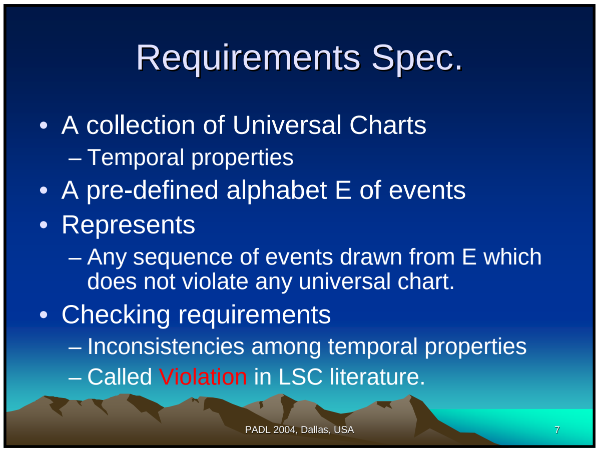
Legal exec of Body

! green
!yellow
? yellow
!red
?green
?red



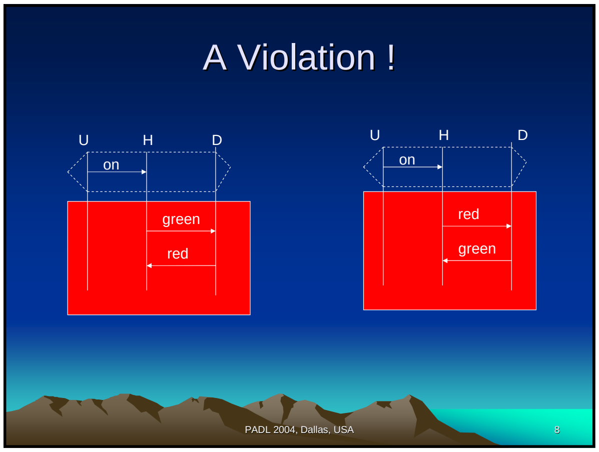
Live Sequence Charts

- Universal Charts (NEW !)
  - In any system behavior …
  - … an exec. of the pre-chart must be eventually followed by an exec. of body chart
- Existential Charts (not discussed here!)
  - There exists a system behavior …
  - … an execution of pre-chart followed by body chart occurs

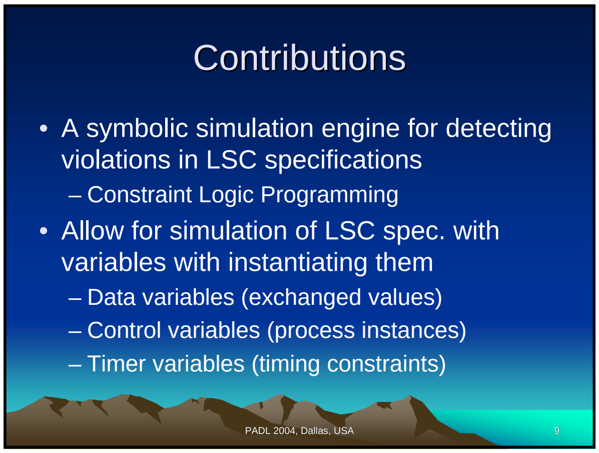


Requirements Spec.

- A collection of Universal Charts
  - Temporal properties
- A pre-defined alphabet E of events
- Represents
  - Any sequence of events drawn from E which does not violate any universal chart.
- Checking requirements
  - Inconsistencies among temporal properties
  - Called Violation in LSC literature.

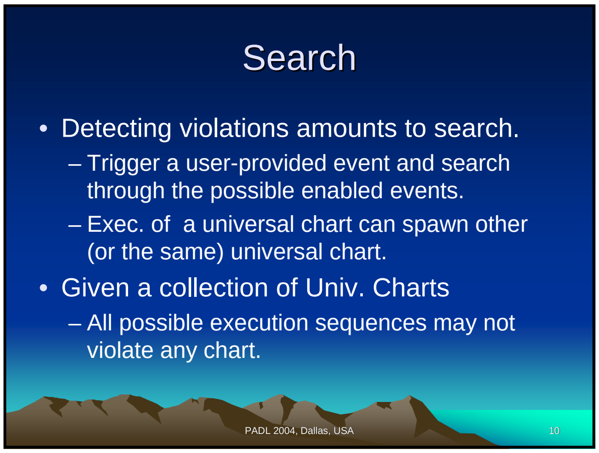


A Violation !



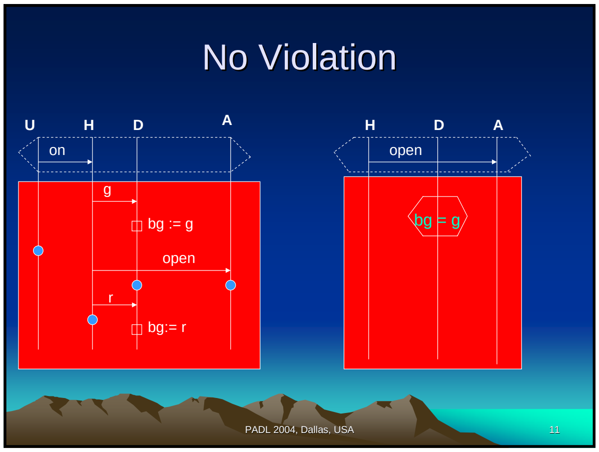
Contributions

- A symbolic simulation engine for detecting violations in LSC specifications
  - Constraint Logic Programming
- Allow for simulation of LSC spec. with variables with instantiating them
  - Data variables (exchanged values)
  - Control variables (process instances)
  - Timer variables (timing constraints)

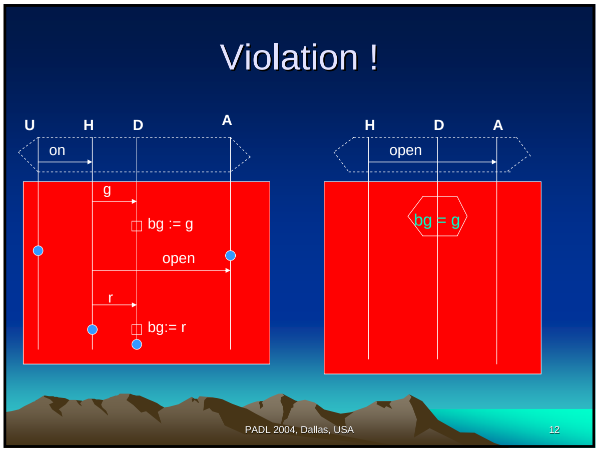


Search

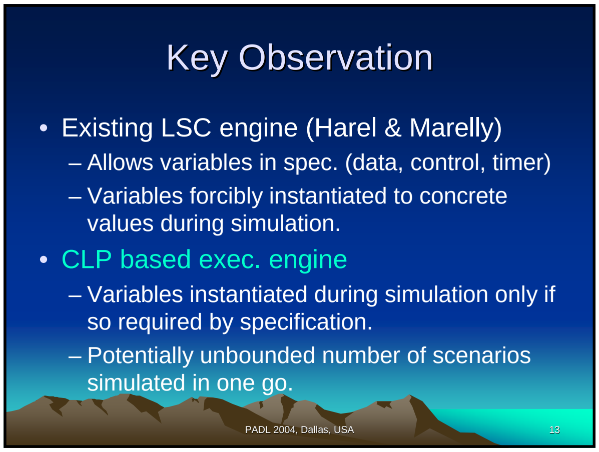
- Detecting violations amounts to search.
  - Trigger a user-provided event and search through the possible enabled events.
  - Exec. of a universal chart can spawn other (or the same) universal chart.
- Given a collection of Univ. Charts
  - All possible execution sequences may not violate any chart.



No Violation

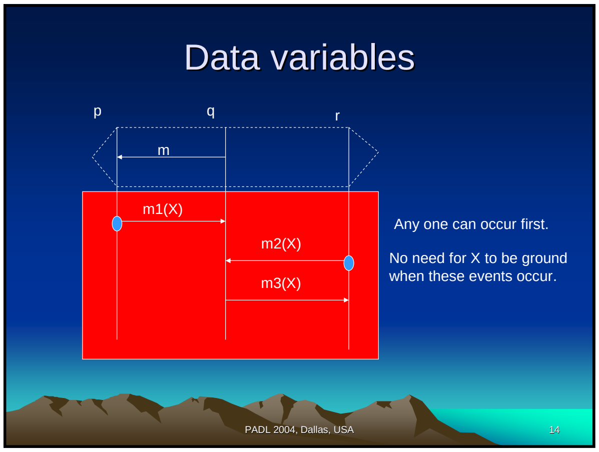


Violation !



Key Observation

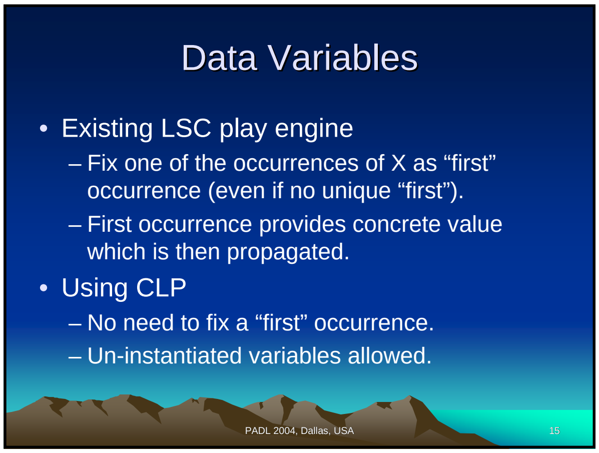
- Existing LSC engine (Harel & Marelly)
  - – Allows variables in spec. (data, control, timer)
  - – Variables forcibly instantiated to concrete values during simulation.
- CLP based exec. engine
  - – Variables instantiated during simulation only if so required by specification.
  - – Potentially unbounded number of scenarios simulated in one go.



Data variables

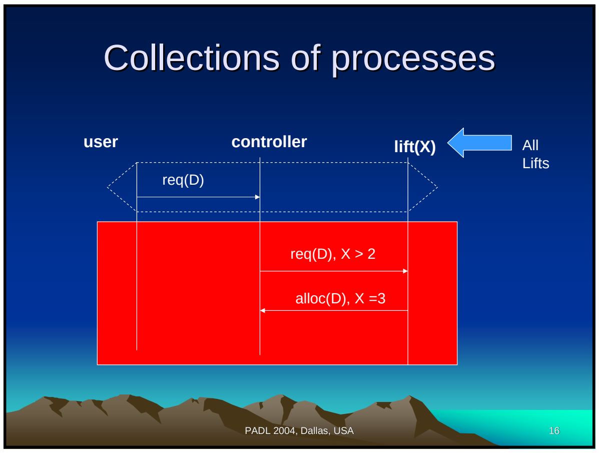
Any one can occur first.

No need for X to be ground when these events occur.



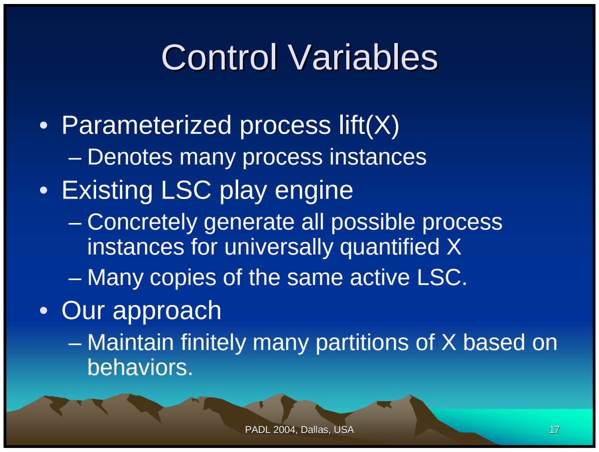
Data Variables

- Existing LSC play engine
  - – Fix one of the occurrences of X as “first” occurrence (even if no unique “first”).
  - – First occurrence provides concrete value which is then propagated.
- Using CLP
  - – No need to fix a “first” occurrence.
  - – Un-instantiated variables allowed.



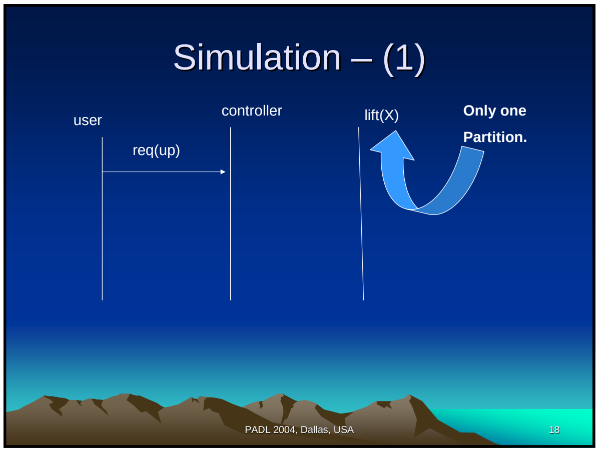
Collections of processes

All Lifts



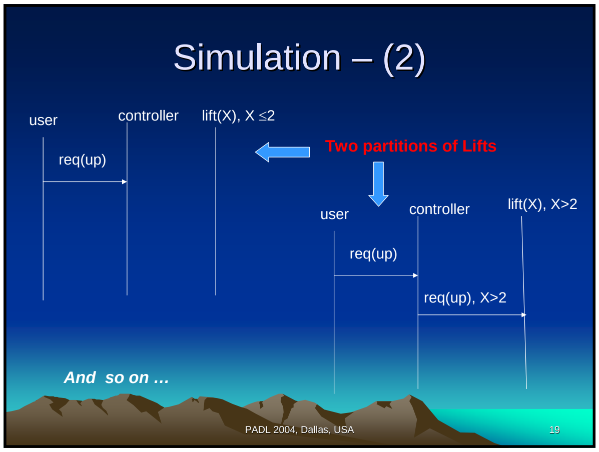
Control Variables

- Parameterized process lift(X)
  - – Denotes many process instances
- Existing LSC play engine
  - – Concretely generate all possible process instances for universally quantified X
  - – Many copies of the same active LSC.
- Our approach
  - – Maintain finitely many partitions of X based on behaviors.



Simulation – (1)

Only one Partition.



Simulation – (2)

user

controller

lift(X), X ≤2

req(up)

Two partitions of Lifts

user

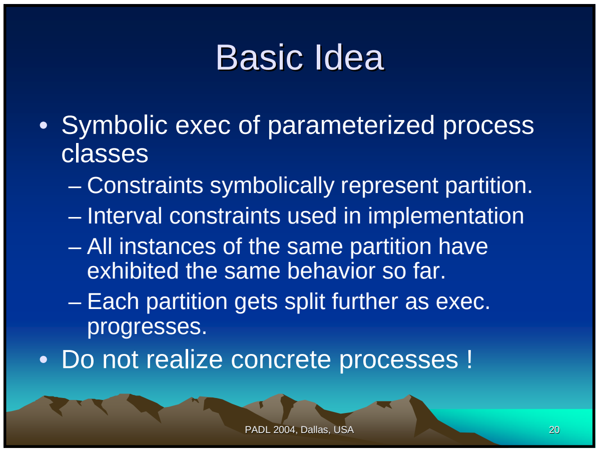
controller

lift(X), X>2

req(up)

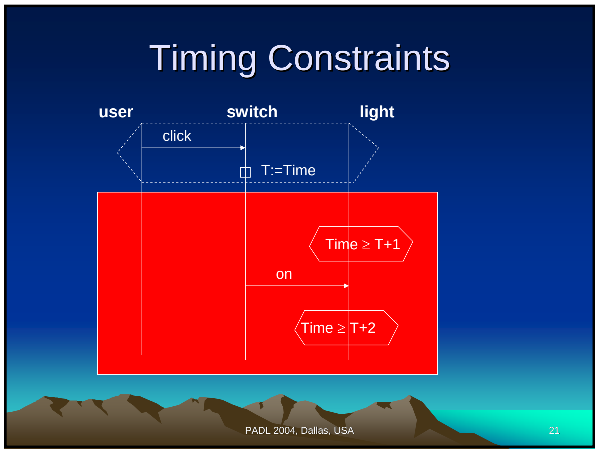
req(up), X>2

And so on …



Basic Idea

- Symbolic exec of parameterized process classes
  - – Constraints symbolically represent partition.
  - – Interval constraints used in implementation
  - – All instances of the same partition have exhibited the same behavior so far.
  - – Each partition gets split further as exec. progresses.
- Do not realize concrete processes !



Timing Constraints

user **switch** **light**

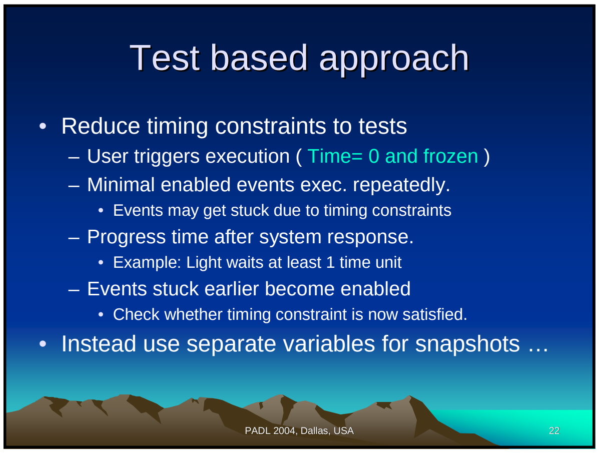
click

T:=Time

Time ≥ T+1

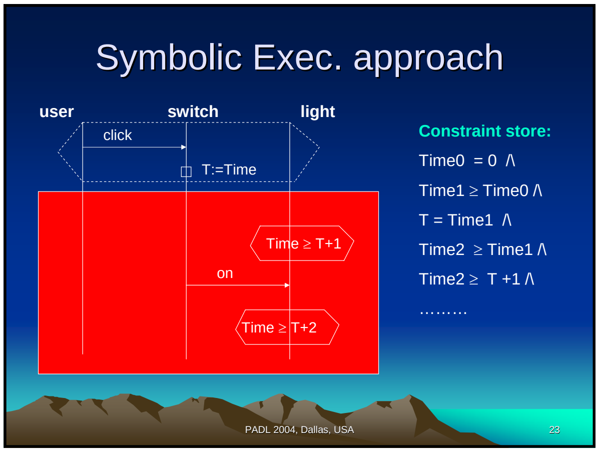
on

Time ≥ T+2



Test based approach

- Reduce timing constraints to tests
  - – User triggers execution (Time= 0 and frozen)
  - – Minimal enabled events exec. repeatedly.
    - Events may get stuck due to timing constraints
  - – Progress time after system response.
    - Example: Light waits at least 1 time unit
  - – Events stuck earlier become enabled
    - Check whether timing constraint is now satisfied.
- Instead use separate variables for snapshots …



Symbolic Exec. approach

user **switch** **light**

click

T:=Time

Time ≥ T+1

on

Time ≥ T+2

Constraint store:

Time0 = 0 /\

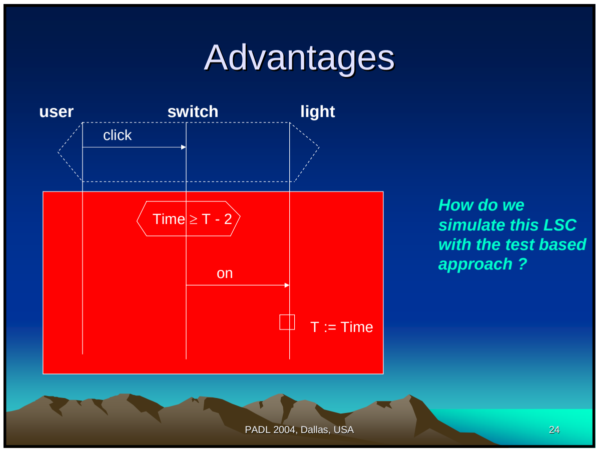
Time1 ≥ Time0 /\

T = Time1 /\

Time2 ≥ Time1 /\

Time2 ≥ T +1 /\

………



Advantages

user **switch** **light**

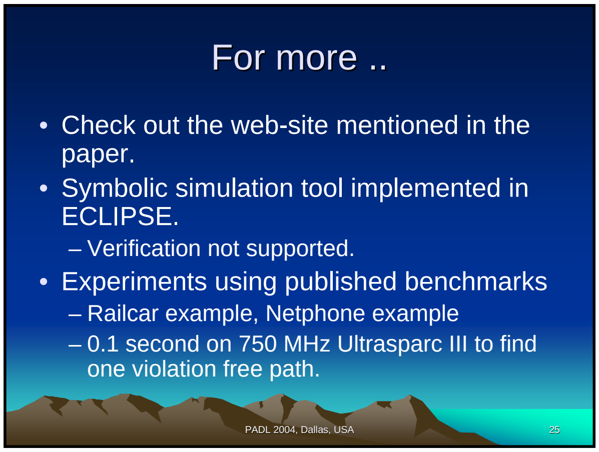
click

Time ≥ T - 2

on

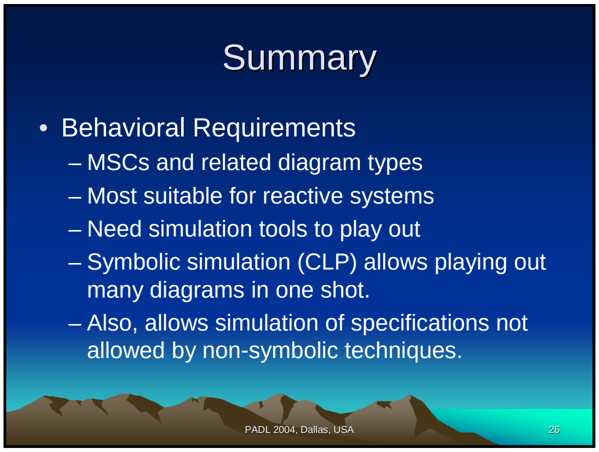
T := Time

How do we simulate this LSC with the test based approach ?



For more ..

- Check out the web-site mentioned in the paper.
- Symbolic simulation tool implemented in ECLIPSE.
  - Verification not supported.
- Experiments using published benchmarks
  - Railcar example, Netphone example
  - 0.1 second on 750 MHz Ultrasparc III to find one violation free path.



Summary

- Behavioral Requirements
  - MSCs and related diagram types
  - Most suitable for reactive systems
  - Need simulation tools to play out
  - Symbolic simulation (CLP) allows playing out many diagrams in one shot.
  - Also, allows simulation of specifications not allowed by non-symbolic techniques.