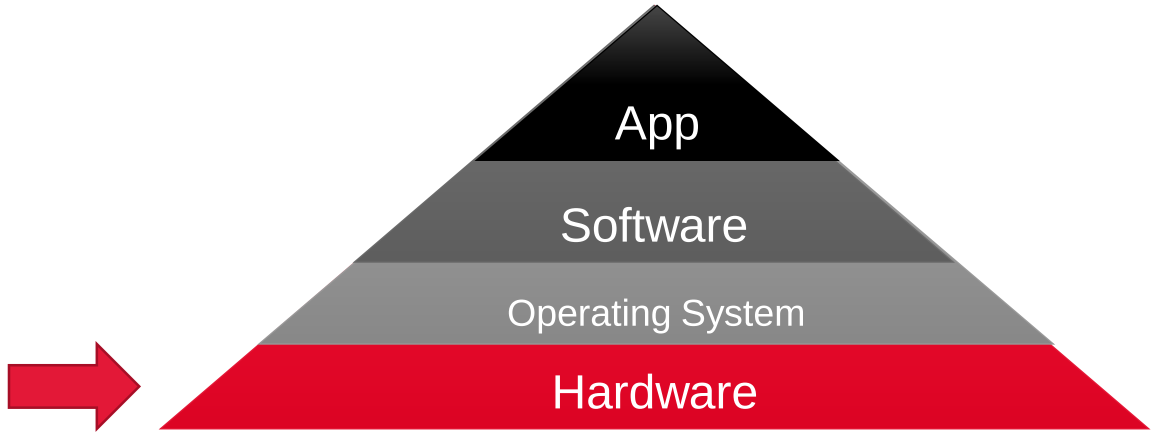
The Role of Formal Verification in Security Assurance

Dan Benua

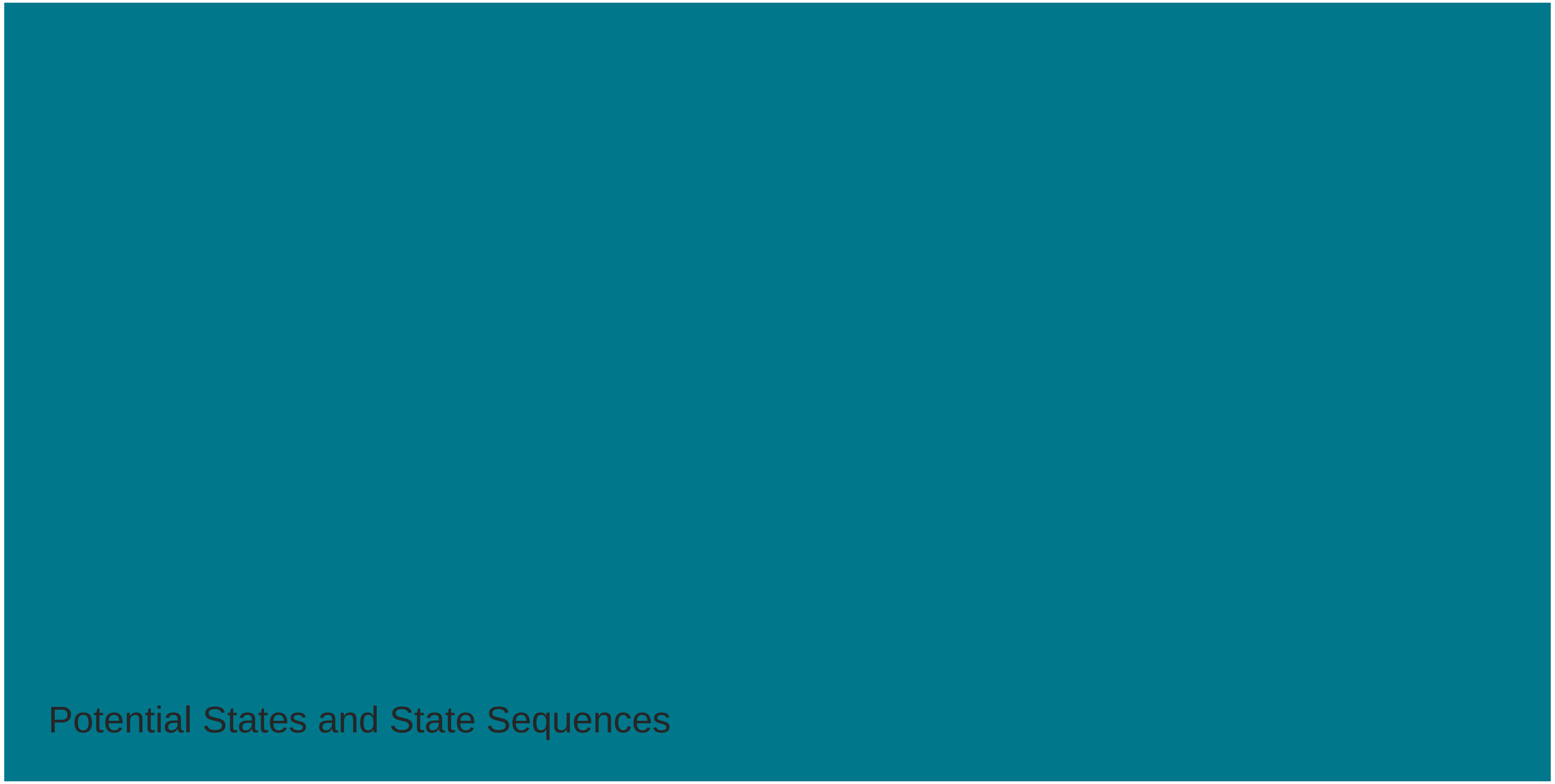
July 2022

Formal Verification – Hardware Security

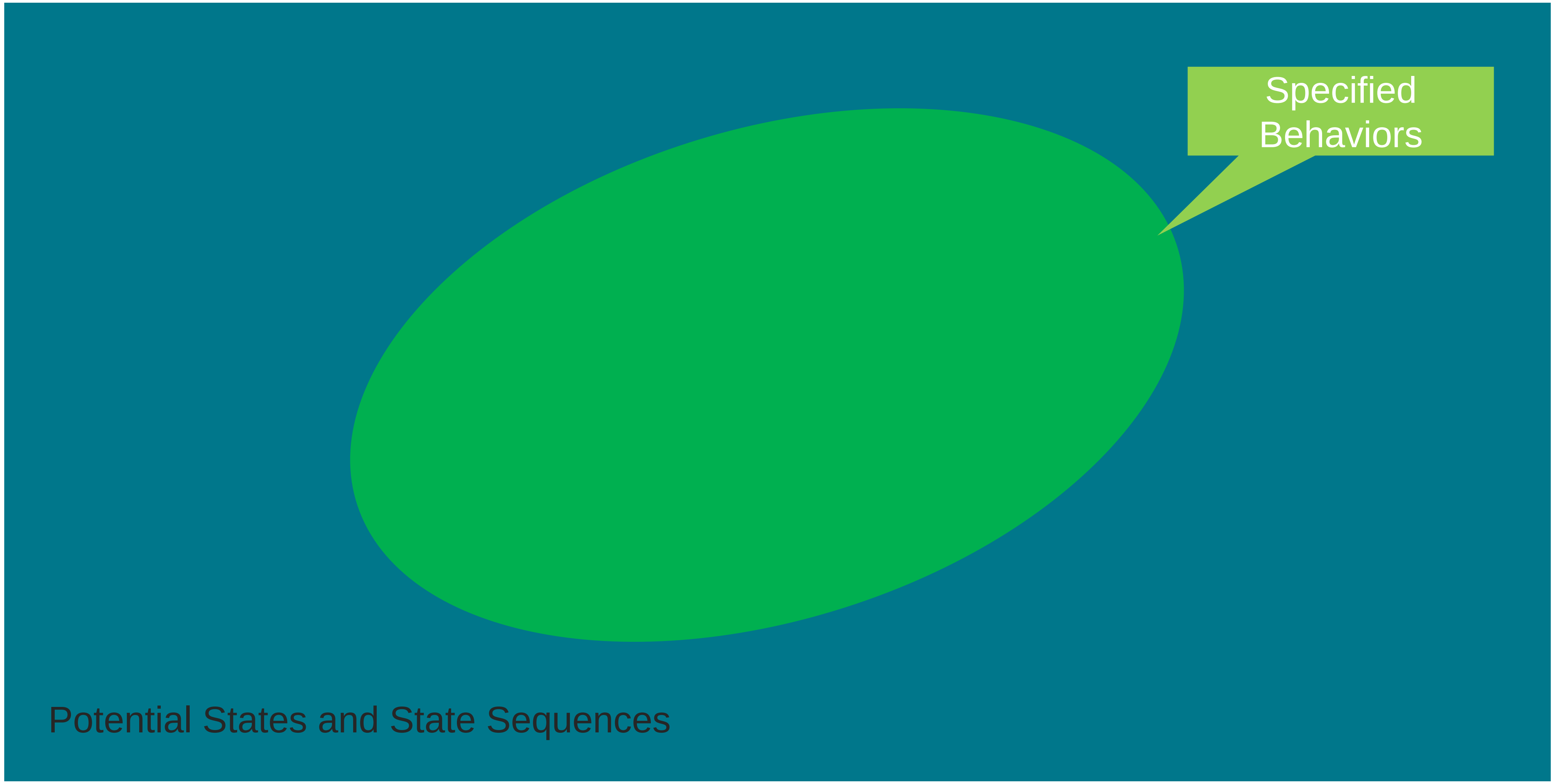


- Exhaustive formal verification can eliminate unspecified behaviors that hackers could exploit.
- Formal is exceptional at finding corner case behaviors, even when exhaustive proofs of correctness are not possible at system-level scale.
- Jasper's specialized apps can be applied to specific known security vulnerabilities and provide results with greater efficiency or completeness compared to alternative methodologies

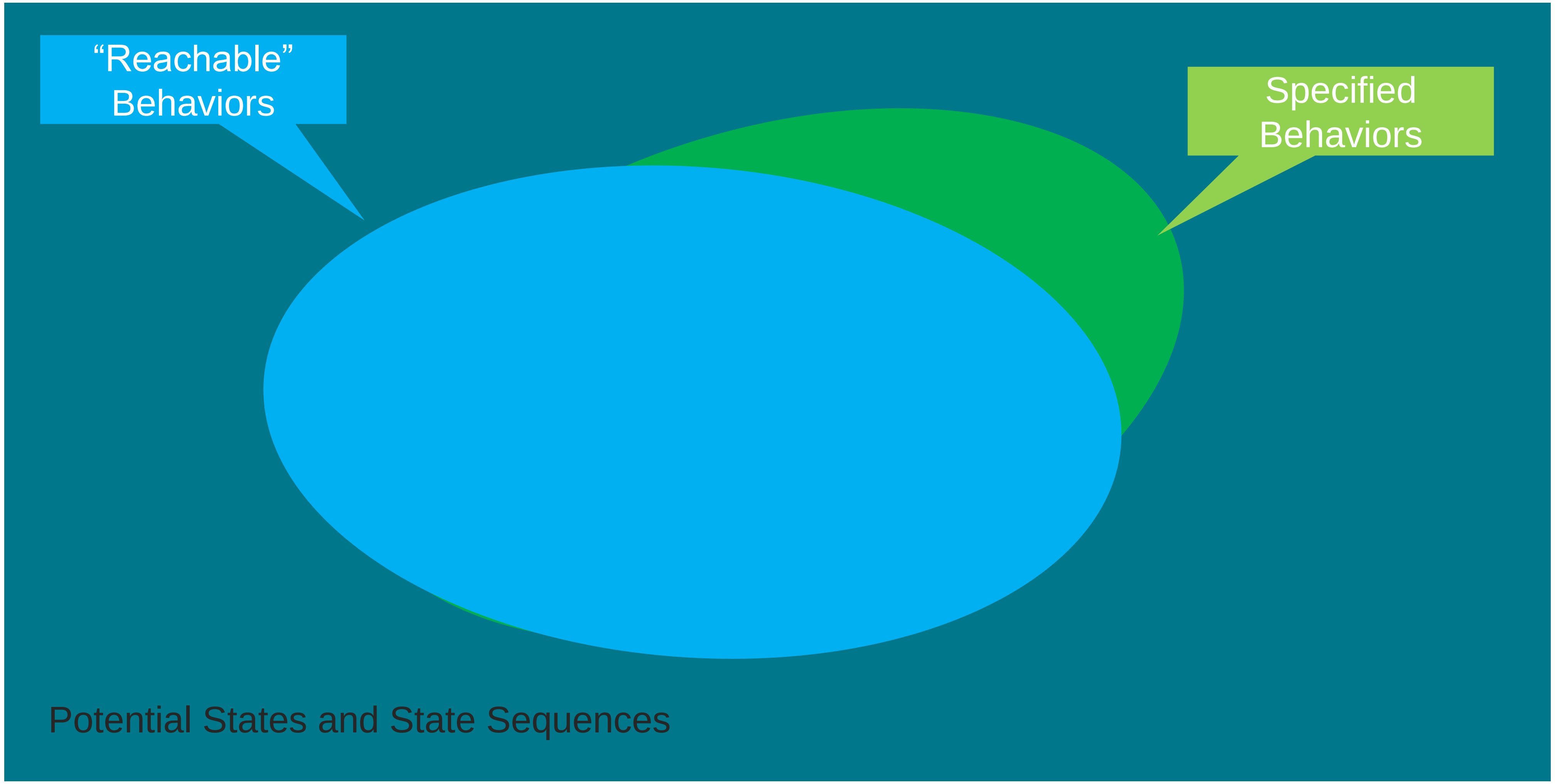
State Space View of Security



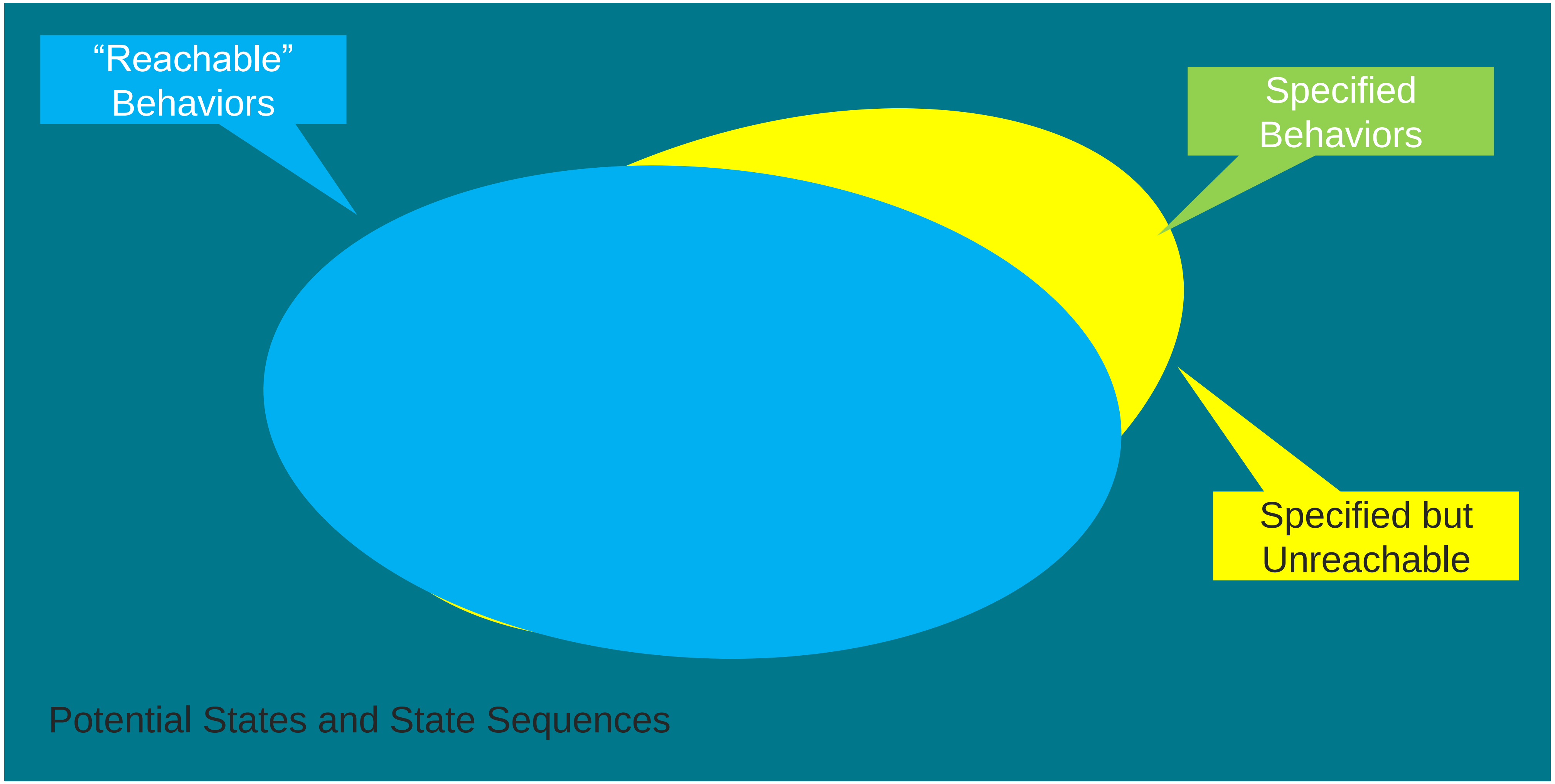
State Space View of Security



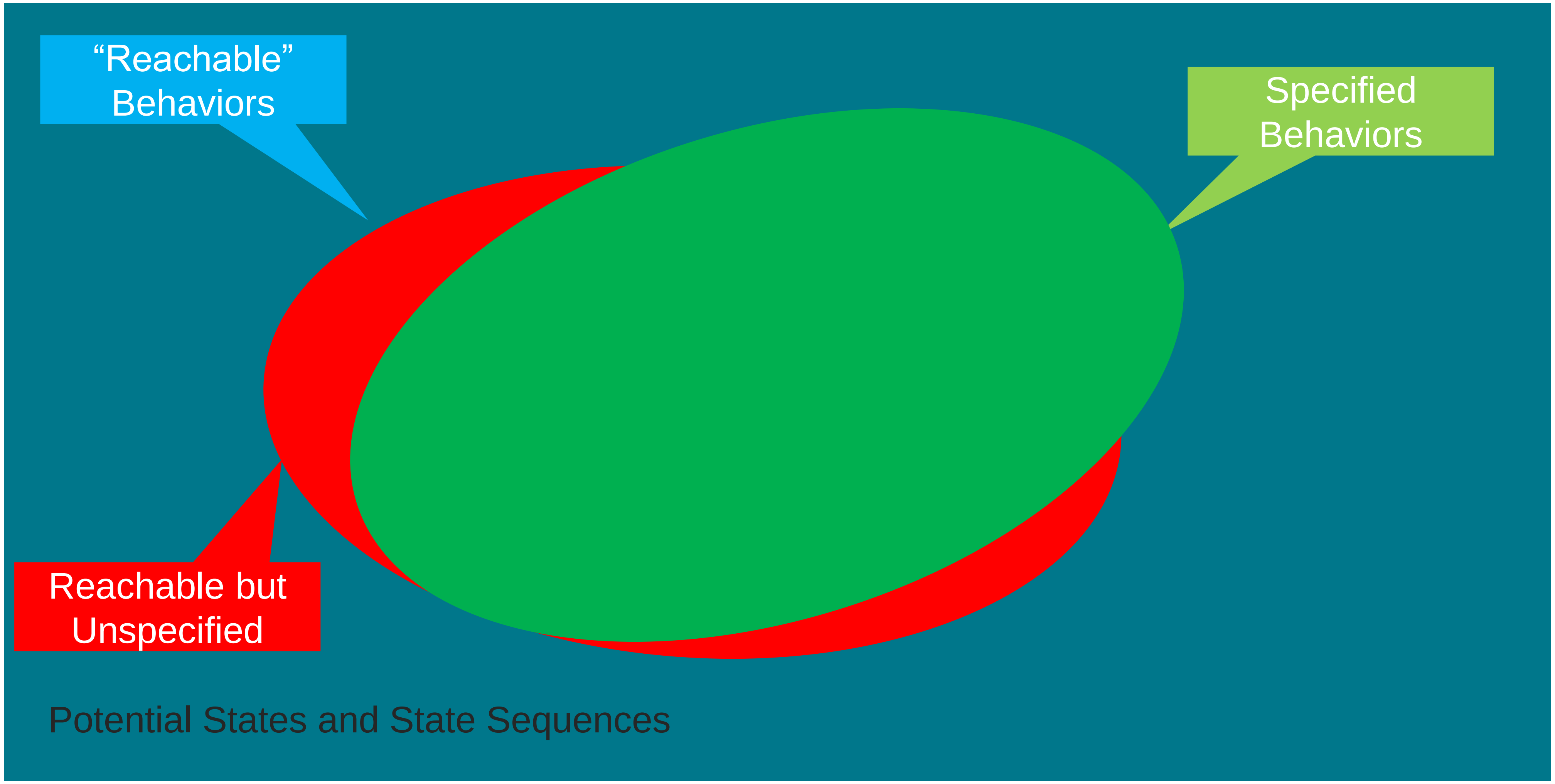
State Space View of Hardware Security



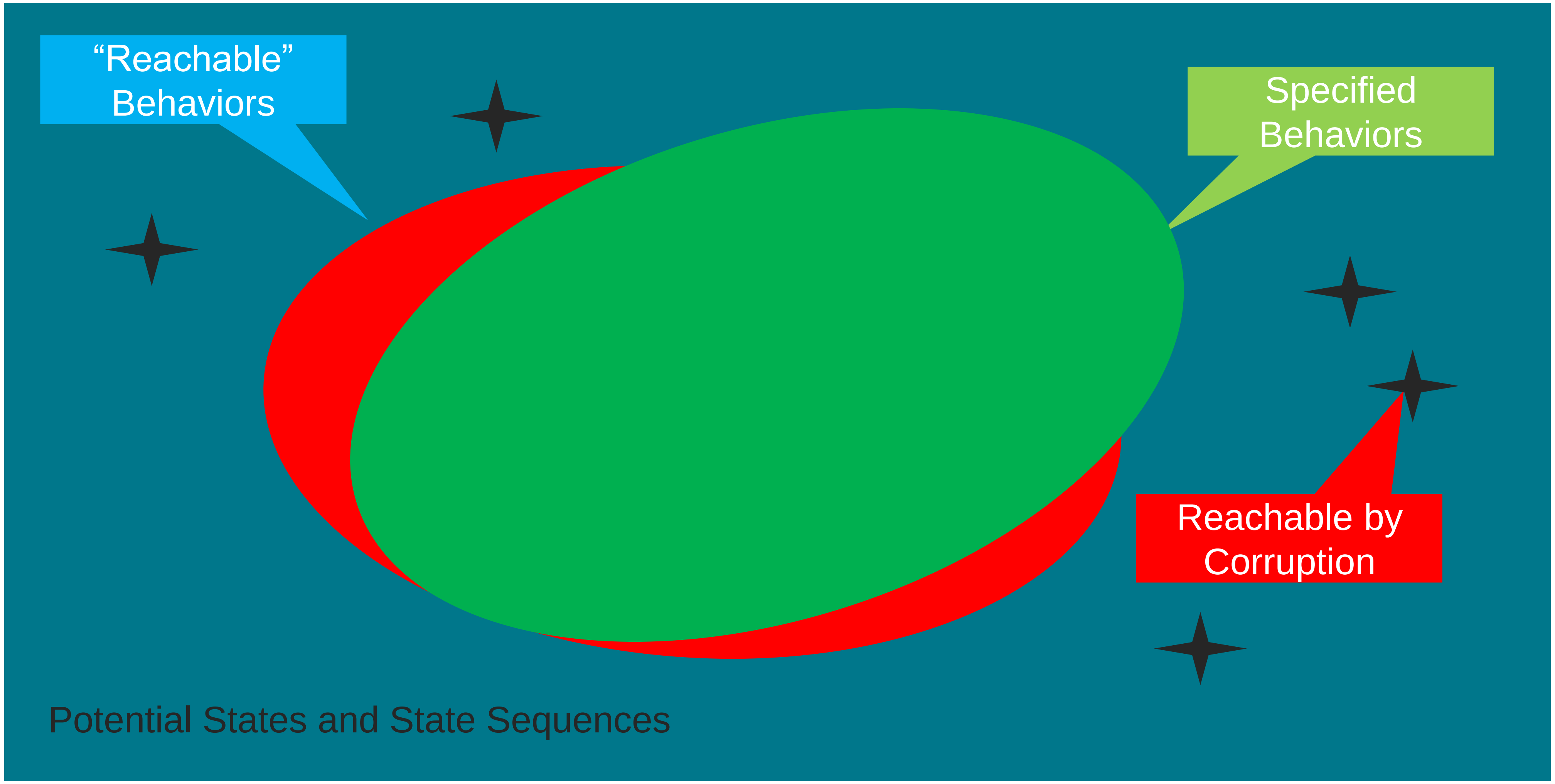
State Space View of Hardware Security


"Reachable" Behaviors
Specified Behaviors
Specified but Unreachable
Potential States and State Sequences

State Space View of Hardware Security


"Reachable" Behaviors
Specified Behaviors
Reachable but Unspecified
Potential States and State Sequences

State Space View of Hardware Security



Hardware Bugs – Security Vulnerabilities

- State Machine Deadlock
- Buffer Overflow
- Incorrect Register Access
- Unexpected X-propagation
- Bus Protocol Violation
- Improper ECO Implementation

Hardware Bugs – Security Vulnerabilities

- State Machine Deadlock
- Buffer Overflow
- Incorrect Register Access
- Unexpected X-propagation
- Bus Protocol Violation
- Improper ECO Implementation

- Denial of Service

Hardware Bugs – Security Vulnerabilities

- State Machine Deadlock
- Buffer Overflow
- Incorrect Register Access
- Unexpected X-propagation
- Bus Protocol Violation
- Improper ECO Implementation

- Denial of Service
- Data Corruption, Unexpected Control Flow

Hardware Bugs – Security Vulnerabilities

- State Machine Deadlock
- Buffer Overflow
- Incorrect Register Access
- Unexpected X-propagation
- Bus Protocol Violation
- Improper ECO Implementation

- Denial of Service
- Data Corruption, Unexpected Control Flow
- Secure Data Leakage or Corruption

Hardware Bugs – Security Vulnerabilities

- State Machine Deadlock
- Buffer Overflow
- Incorrect Register Access
- Unexpected X-propagation
- Bus Protocol Violation
- Improper ECO Implementation

- Denial of Service
- Data Corruption, Unexpected Control Flow
- Secure Data Leakage or Corruption
- Data Corruption, Unexpected Control Flow

Hardware Bugs – Security Vulnerabilities

• State Machine Deadlock	• Denial of Service
• Buffer Overflow	• Data Corruption, Unexpected Control Flow
• Incorrect Register Access	• Secure Data Leakage or Corruption
• Unexpected X-propagation	• Data Corruption, Unexpected Control Flow
• Bus Protocol Violation	• Data Corruption
• Improper ECO Implementation	

Hardware Bugs – Security Vulnerabilities

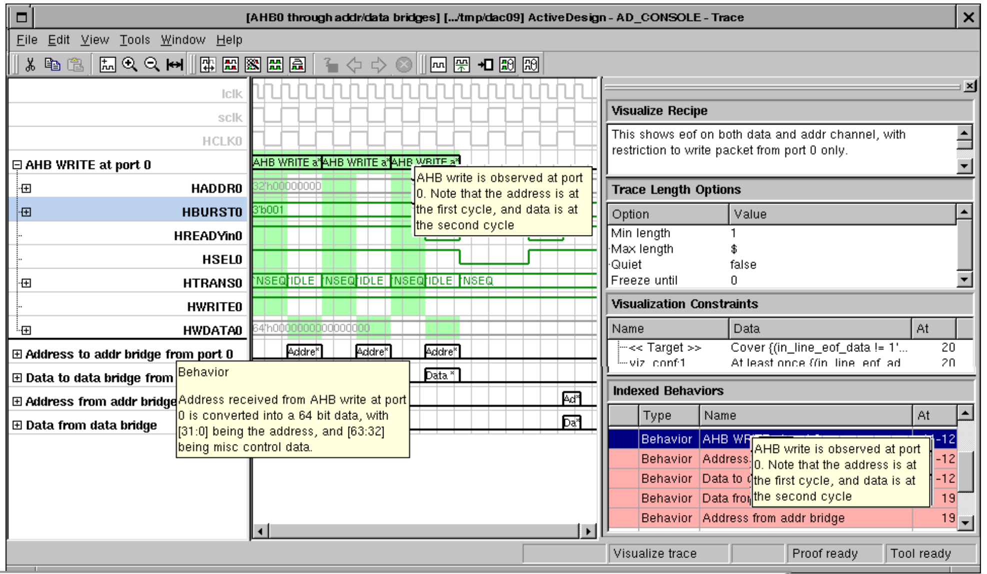
• State Machine Deadlock	• Denial of Service
• Buffer Overflow	• Data Corruption, Unexpected Control Flow
• Incorrect Register Access	• Secure Data Leakage or Corruption
• Unexpected X-propagation	• Data Corruption, Unexpected Control Flow
• Bus Protocol Violation	• Data Corruption
• Improper ECO Implementation	• Vulnerability Insertion

Jasper: Formal Verification Platform

Solve **specific verification problems** with targeted Jasper Apps

Formal Property Verification App	SuperLint (AFL) App	Design Coverage Verification App	Sequential Equivalence Checking App
X-Propagation Verification App	Control/Status Register Verif. App	Connectivity Verification App	Coverage Unreachability App
Clock Domain Crossing App	Functional Safety Verification App	Low Power Verification App	Security Path Verification App

Highly interactive **formal debug** transforms to fit the App



AHB write is observed at port 0. Note that the address is at the first cycle, and data is at the second cycle

Behavior

Address received from AHB write at port 0 is converted into a 64 bit data, with [31:0] being the address, and [63:32] being misc control data.

Visualize Recipe

This shows eof on both data and addr channel, with restriction to write packet from port 0 only.

Trace Length Options

Option	Value
Min length	1
Max length	$
Quiet	false
Freeze until	0

Visualization Constraints

Name	Data	At
<< Target >>	Cover {(in_line_eof_data != 1'...	20
viz_conf1	At least once {(in_line_eof_ad	20

Indexed Behaviors

Type	Name	At
Behavior	AHB WR	-12
Behavior	Address	-12
Behavior	Data to d	-12
Behavior	Data fro	19
Behavior	Address from addr bridge	19

Broad **formal engine** and infrastructure

Assertion Based Verification IPs for AMBA and other common protocols

Programmable Interface via TCL

ProofGrid™ Manager assigns best engine for task

cādence®

CWE Mapping to Formal Apps

CWE	Description	Formal App
1245	Improper Finite State Machines (FSMs) in Hardware Logic	**Superlint**
1247	Missing or Improperly Implemented Protection Against Voltage and Clock Glitches	**CDC**
1271	Uninitialized Value on Reset for Registers Holding Security Settings	**XProp**
1263	Improper Physical Access Control	**SPV**
1282	Assumed-Immutable Data is Stored in Writable Memory	**SPV**
1258	Exposure of Sensitive System Information Due to Uncleared Debug Information	**SPV**
1330	Remanent Data Readable after Memory Erase	**SPV**
1231	Improper Implementation of Lock Protection Registers	**CSR**
1234	Hardware Internal or Debug Modes Allow Override of Locks	**CSR**
1283	Mutable attestation or measurement reporting data	**CSR**
1242	Inclusion of undocumented features	**CSR**
1234	Failure to Disable Reserved Bits	**CSR**
1258	Exposure of Sensitive System Information Due to Uncleared Debug Information	**FPV and SPV**
1262	Improper Access Control for Register Interface	**FPV**
1261	Improper Handling of Single Even Upsets	**FSV**

Why Formal?

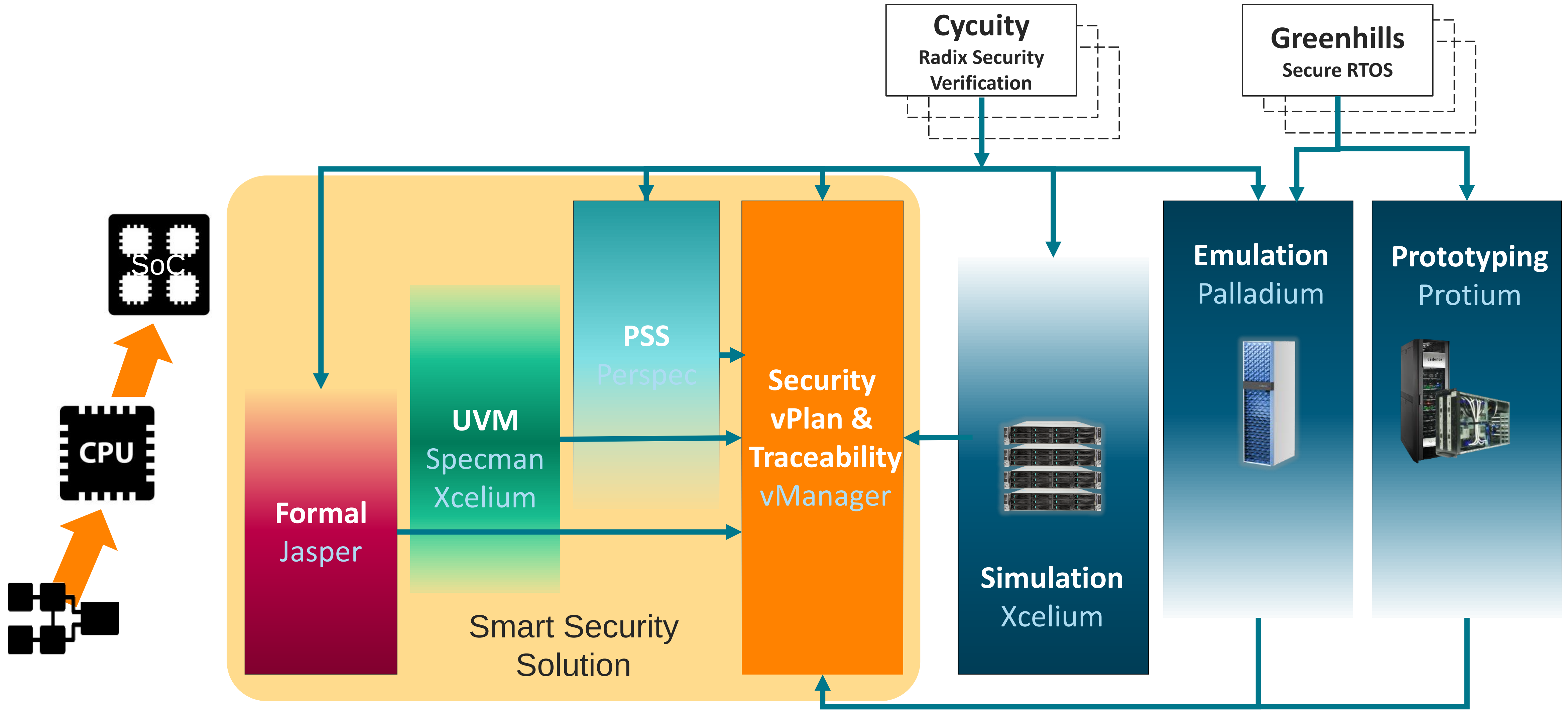
- Exhaustive (in some cases!)
- Exposes Specification Gaps and Ambiguities
- Natural Emphasis on Negative Testing (Counter-Examples)
- Tight Integration of Functional and Security Verification
- Ability to model taint propagation

Why Formal?

- Exhaustive (in some cases!)
- Exposes Specification Gaps and Ambiguities
- Natural Emphasis on Negative Testing (Counter-Examples)
- Tight Integration of Functional and Security Verification
- Ability to model taint propagation

But What About Capacity?

Cadence Security Verification Solution and Partners



Conclusions

- Formal is a key component of hardware security verification
- Security verifications start with basic functional verification and focusses on negative testing.
- Formal must be integrated with other components of the security verification environment.
- If formal is used early in design bring-up it can set the stage for later focused vulnerability analysis.