

# Bringing Formal Property Verification Methodology to an ASIC Design

Erik Seligman, Ram Koganti, Kapilan Maheswaran, Rami Naqib

Advanced Components Division (ACD),  
Digital Enterprise Group, Intel  
Corporation

February 23, 2006



# Purpose

- Formal Property Verification (FPV):  
important for ASIC validation
  - Assists design process
  - Finds bugs
  - Increases overall confidence
- Share FPV BKM's with other ASIC teams
  - Many teams working in isolation
  - Need more industry-wide discussion
    - *Methodology, not just tools*

# Definitions

- **Assertion**
  - Statement that must be true in all cases.
- **Assumption**
  - Assertion treated as always-true constraint for formal proofs.
- **Formal Property Verification (FPV)**
  - Mathematical proofs, not simulation
  - Proves assertions: all possible test vectors

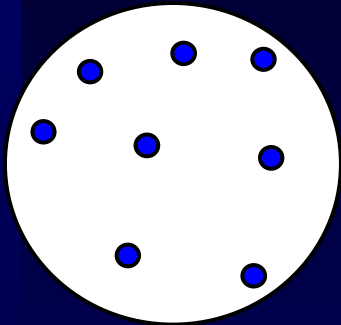
# Agenda

- Motivation: Why Formal Property Verification?
- FPV Methodology in Blackford
- Case Studies
- Results & Recommendations
- Conclusions, Q&A

# Agenda

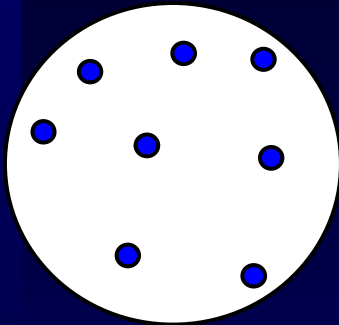
- *Motivation: Why Formal Property Verification?*
- FPV Methodology in Blackford
- Case Studies
- Results & Recommendations
- Conclusions, Q&A

# Motivation for Formal Property Verification

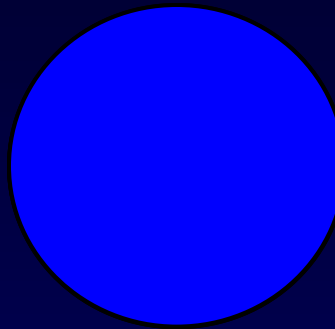


Simulation:  
spot  
coverage of  
design space

# Motivation for Formal Property Verification

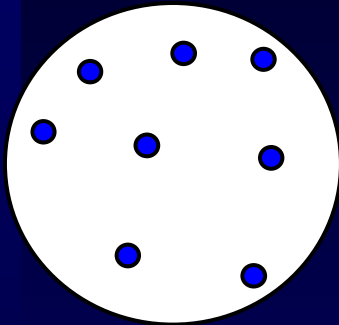


Simulation:  
spot  
coverage of  
design space

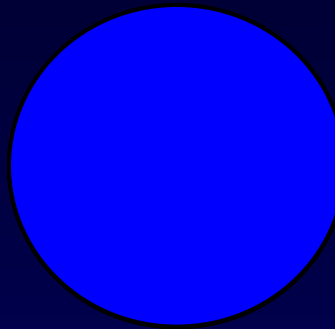


Formal  
Property  
Verification  
(ideal case):  
full coverage  
of design  
space

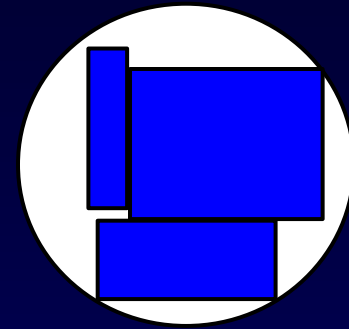
# Motivation for Formal Property Verification



Simulation:  
spot  
coverage of  
design space



Formal  
Property  
Verification  
(ideal case):  
full coverage  
of design  
space



Formal  
Property  
Verification  
(real life):  
full coverage  
in some  
areas

# Major Benefits of FPV for ASIC Projects

- Improving Design Process

# Major Benefits of FPV for ASIC Projects

- Improving Design Process
  - Force Designer to Think Through Logic

# Major Benefits of FPV for ASIC Projects

- Improving Design Process
  - Force Designer to Think Through Logic
  - Help Identify Hidden Assumptions

# Major Benefits of FPV for ASIC Projects

- Improving Design Process
  - Force Designer to Think Through Logic
  - Help Identify Hidden Assumptions
- Bug Hunting

# Major Benefits of FPV for ASIC Projects

- Improving Design Process
  - Force Designer to Think Through Logic
  - Help Identify Hidden Assumptions
- Bug Hunting
  - Unit-Level Validation (before testbench)

# Major Benefits of FPV for ASIC Projects

- Improving Design Process
  - Force Designer to Think Through Logic
  - Help Identify Hidden Assumptions
- Bug Hunting
  - Unit-Level Validation (before testbench)
  - Find Corner Cases Missed in Simulation

# Major Benefits of FPV for ASIC Projects

- Improving Design Process
  - Force Designer to Think Through Logic
  - Help Identify Hidden Assumptions
- Bug Hunting
  - Unit-Level Validation (before testbench)
  - Find Corner Cases Missed in Simulation
  - Quickly Verify Design Changes

# Major Benefits of FPV for ASIC Projects

- Improving Design Process
  - Force Designer to Think Through Logic
  - Help Identify Hidden Assumptions
- Bug Hunting
  - Unit-Level Validation (before testbench)
  - Find Corner Cases Missed in Simulation
  - Quickly Verify Design Changes
- “Peace of Mind”

# Agenda

- Motivation: Why Formal Property Verification?
- *FPV Methodology in Blackford*
- Case Studies
- Results & Recommendations
- Conclusions, Q&A

# Blackford: Starting Point

- FPV seen as too hard in ACD before Blackford
  - “Something for PhDs to work on”
  - Engine tuning, advanced options required
  - Specialized language (ForSpec)

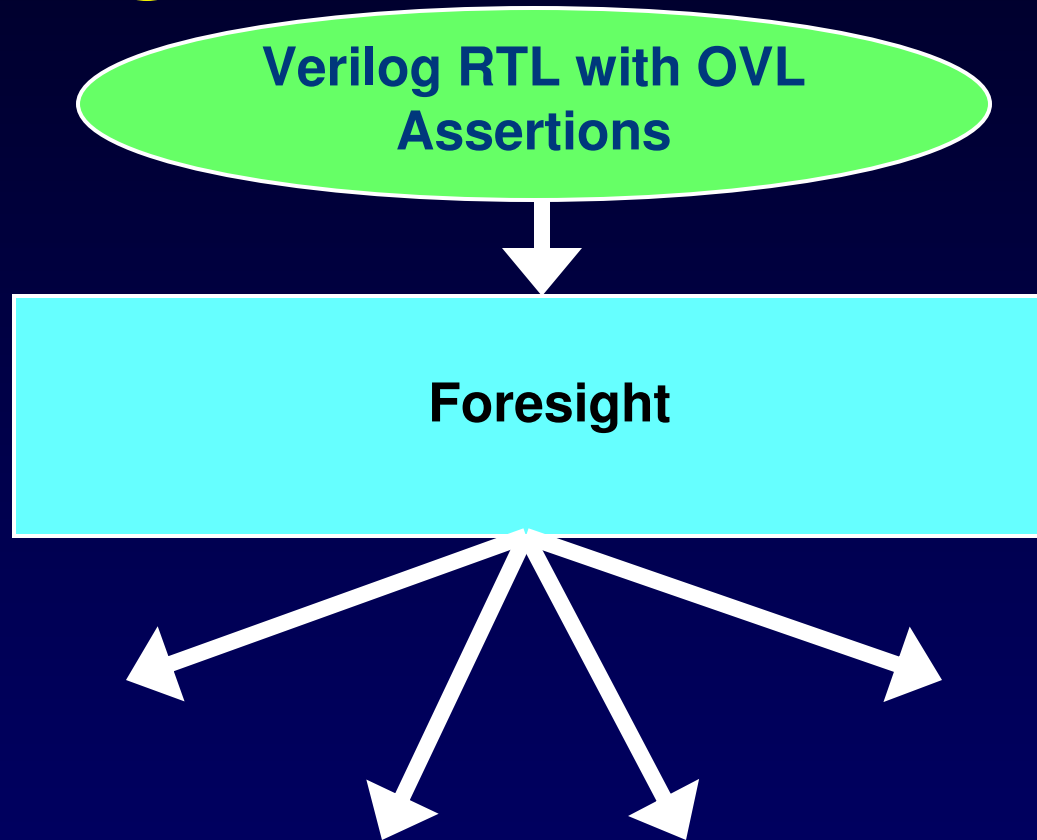
# Blackford: Starting Point

- FPV seen as too hard in ACD before Blackford
  - “Something for PhDs to work on”
  - Engine tuning, advanced options required
  - Specialized language (ForSpec)
- Assertion usage established in ACD RTL
  - Used OVL (Open Verification Library)
    - *See [www.eda.org/ovl](http://www.eda.org/ovl)*

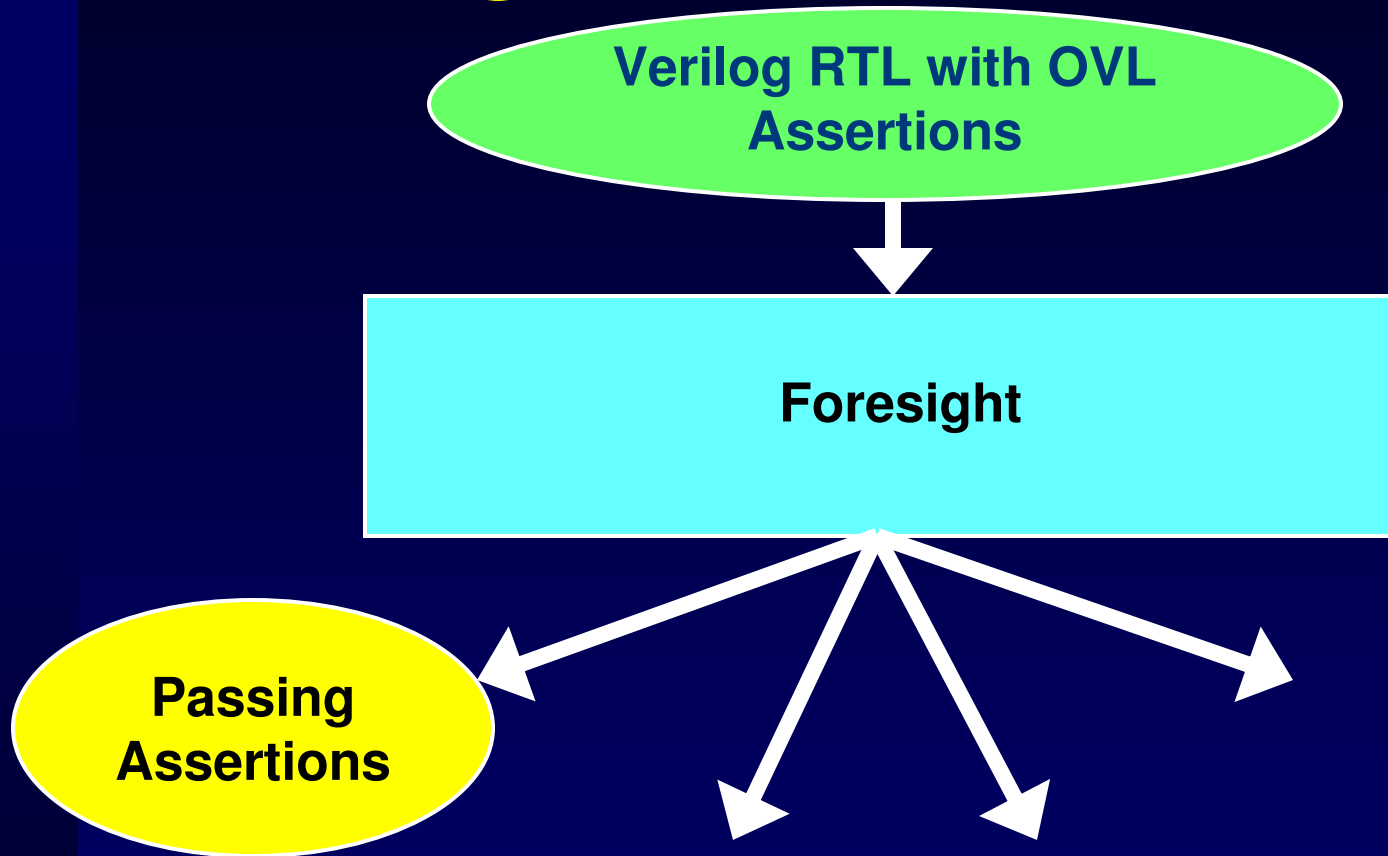
# Blackford: Starting Point

- FPV seen as too hard in ACD before Blackford
  - “Something for PhDs to work on”
  - Engine tuning, advanced options required
  - Specialized language (ForSpec)
- Assertion usage established in ACD RTL
  - Used OVL (Open Verification Library)
    - See [www.eda.org/ovl](http://www.eda.org/ovl)
- Blackford + Intel’s Design Technology team → new “Foresight” tool
  - Push-Button Verification
  - OVL assertions

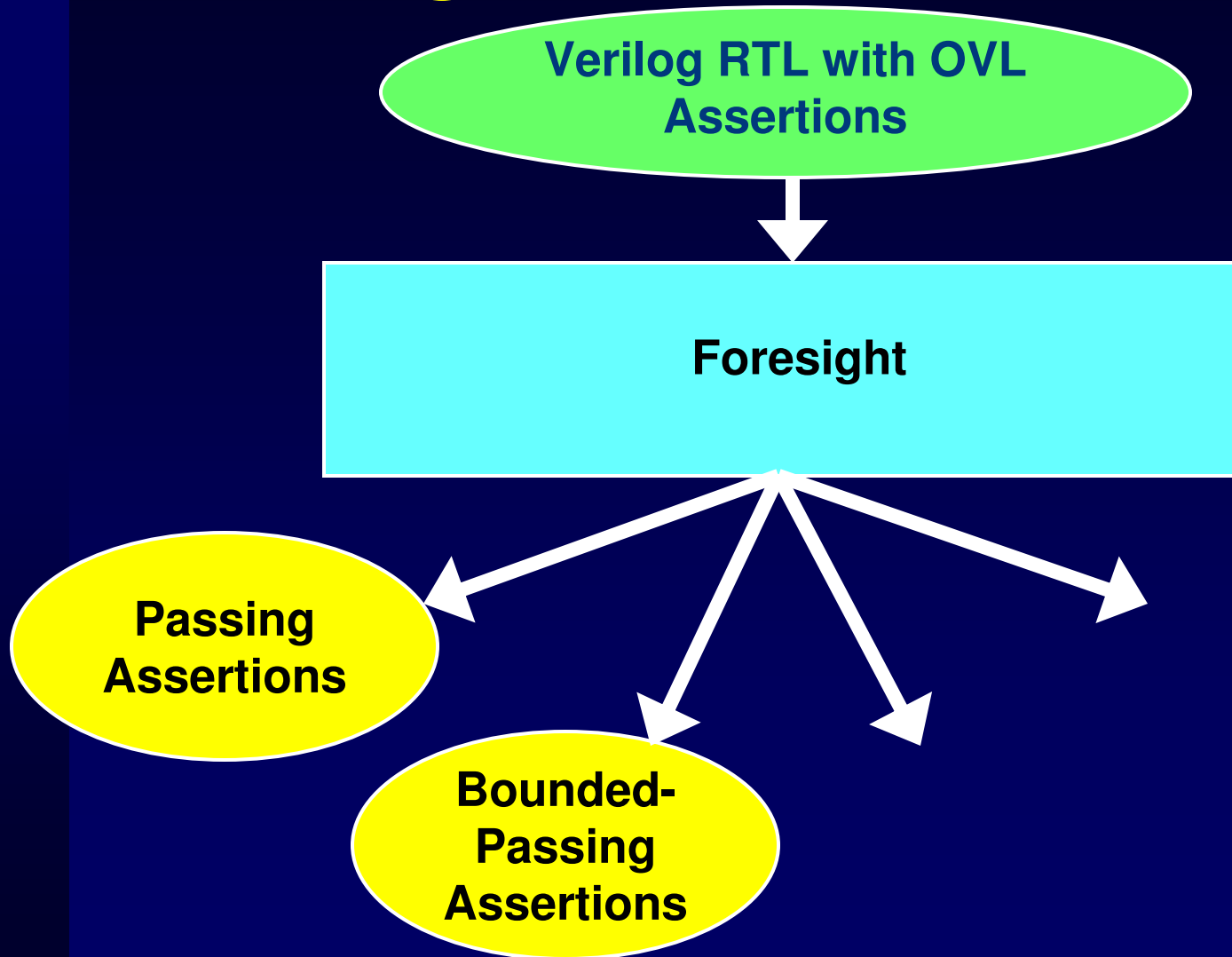
# Foresight: User-Friendly FPV



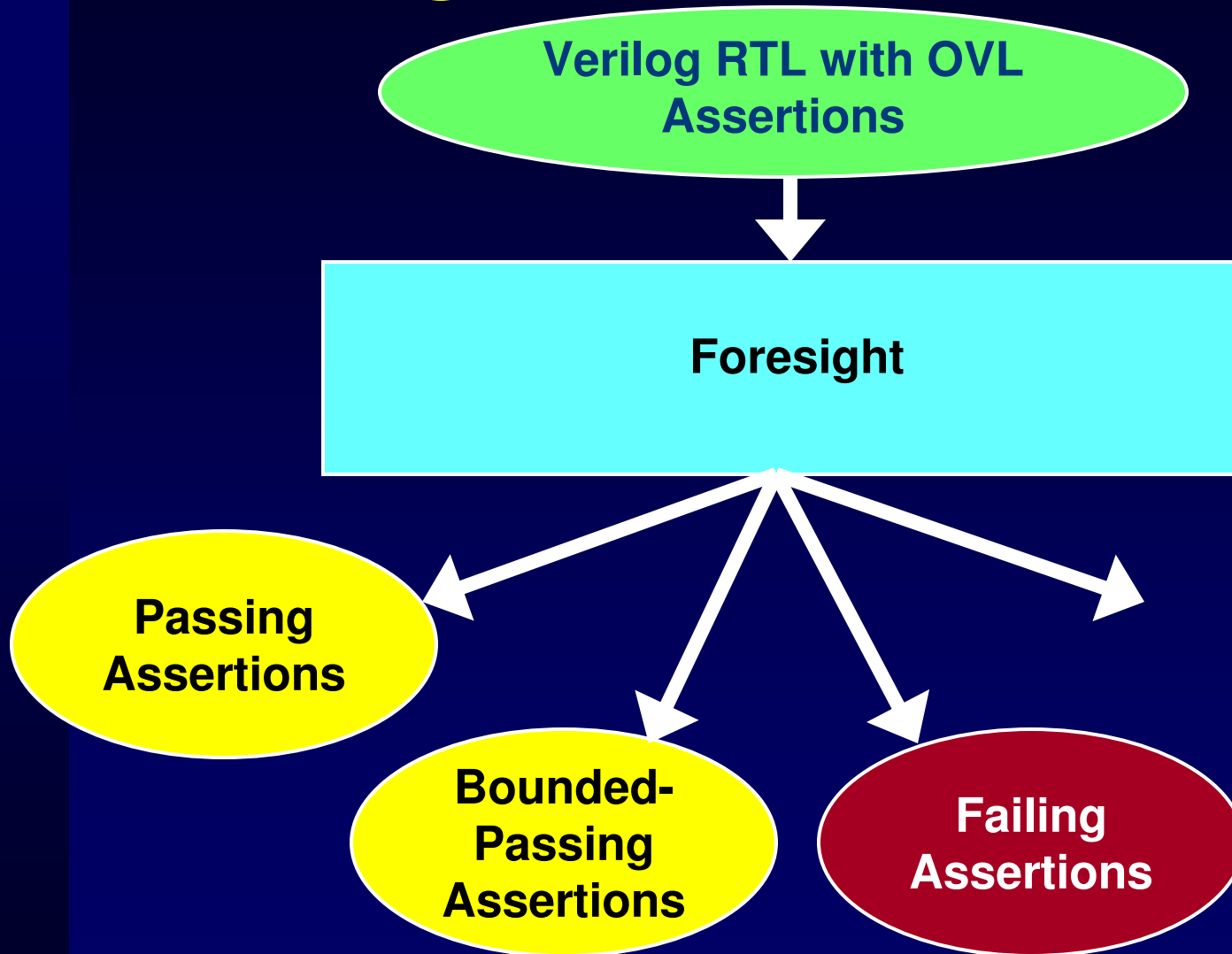
# Foresight: User-Friendly FPV



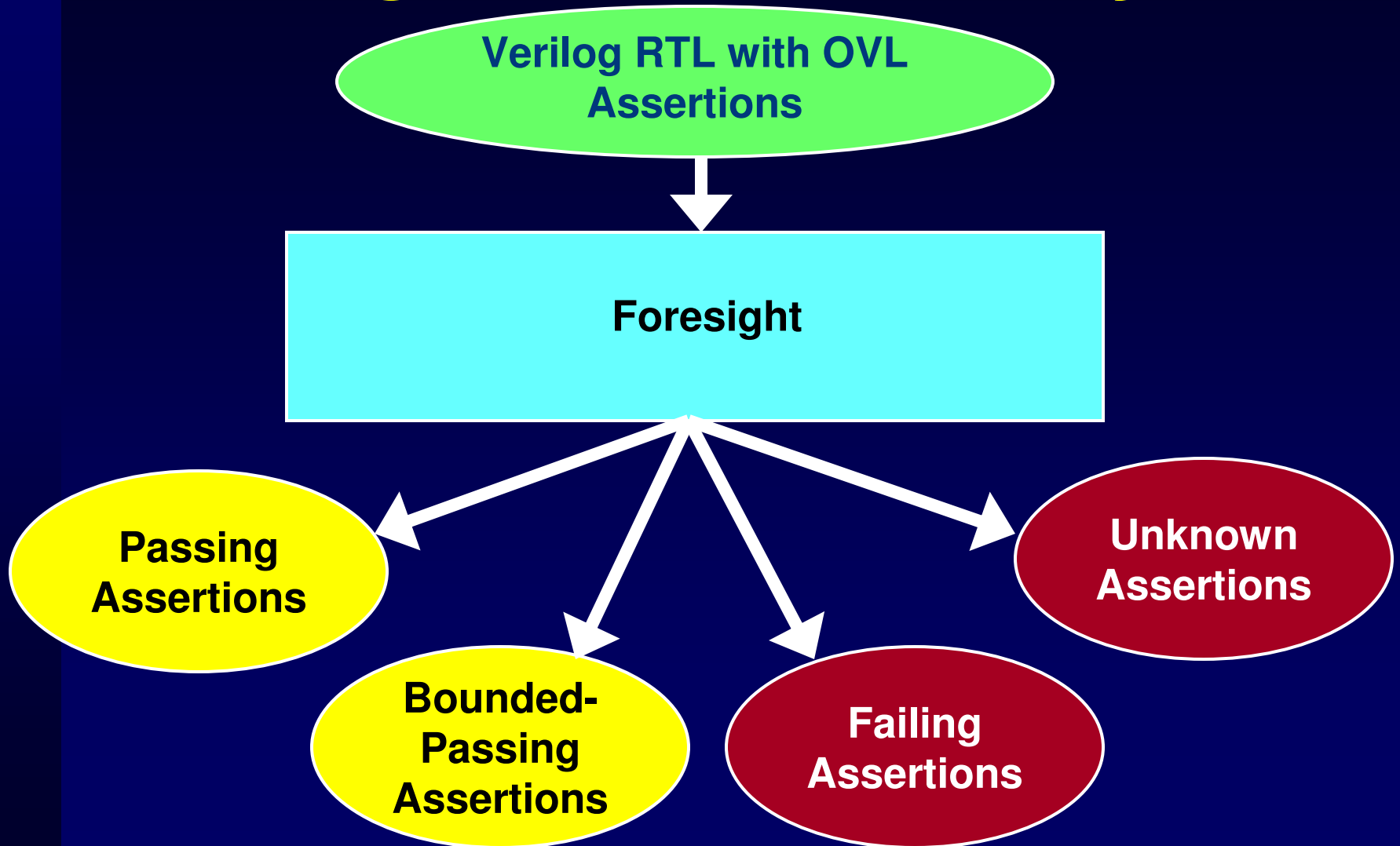
# Foresight: User-Friendly FPV



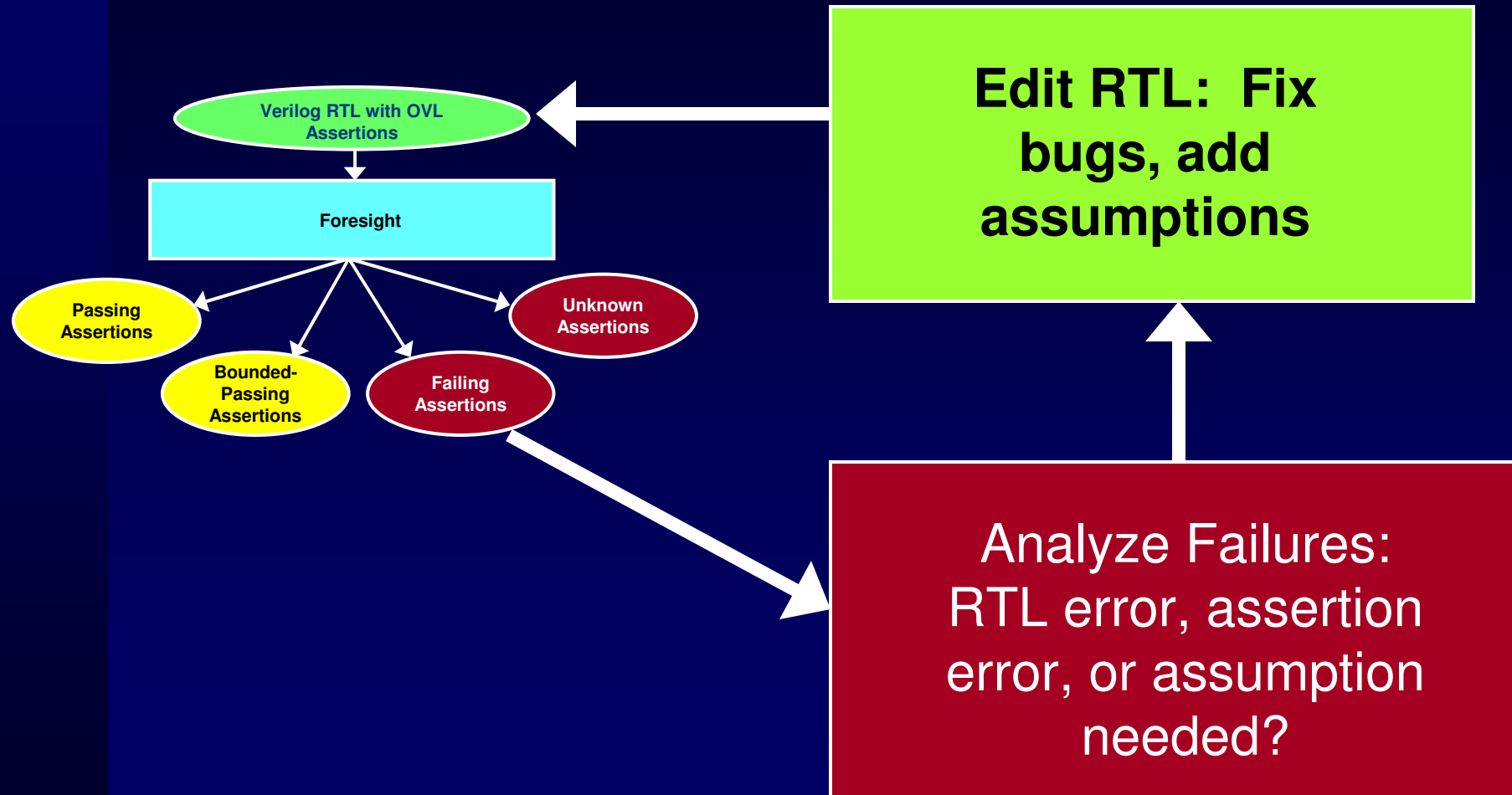
# Foresight: User-Friendly FPV



# Foresight: User-Friendly FPV



# FPV Debug Loop



# Blackford FPV Usage Model

- DEs encouraged to run FPV
  - Purely optional
  - Central owner / support added later

# Blackford FPV Usage Model

- DEs encouraged to run FPV
  - Purely optional
  - Central owner / support added later
- Weekly regression run
  - Detect assertion-violating changes

# Blackford FPV Usage Model

- DEs encouraged to run FPV
  - Purely optional
  - Central owner / support added later
- Weekly regression run
  - Detect assertion-violating changes
- “Property Push”
  - Used assertion coverage tool (0in)
  - Concentrated on at-risk cluster
  - Drove new assertion creation and FPV

# Agenda

- Motivation: Why Formal Property Verification?
- FPV Methodology in Blackford
- *Case Studies*
- Results & Recommendations
- Conclusions, Q&A

# Case Study #1: Finding a Bug

- In one unit, many assertions failed
  - Mid-transaction address changes

# Case Study #1: Finding a Bug

- In one unit, many assertions failed
  - Mid-transaction address changes
- Needed assumption on inputs

```
module BMpeTop;  
  input IssueTxn, CfgAddress, TxnSent;  
  ...  
  assert_win_unchange ... (...IssueTxn, CfgAddress,  
  TxnSent);  
  ...  
endmodule
```

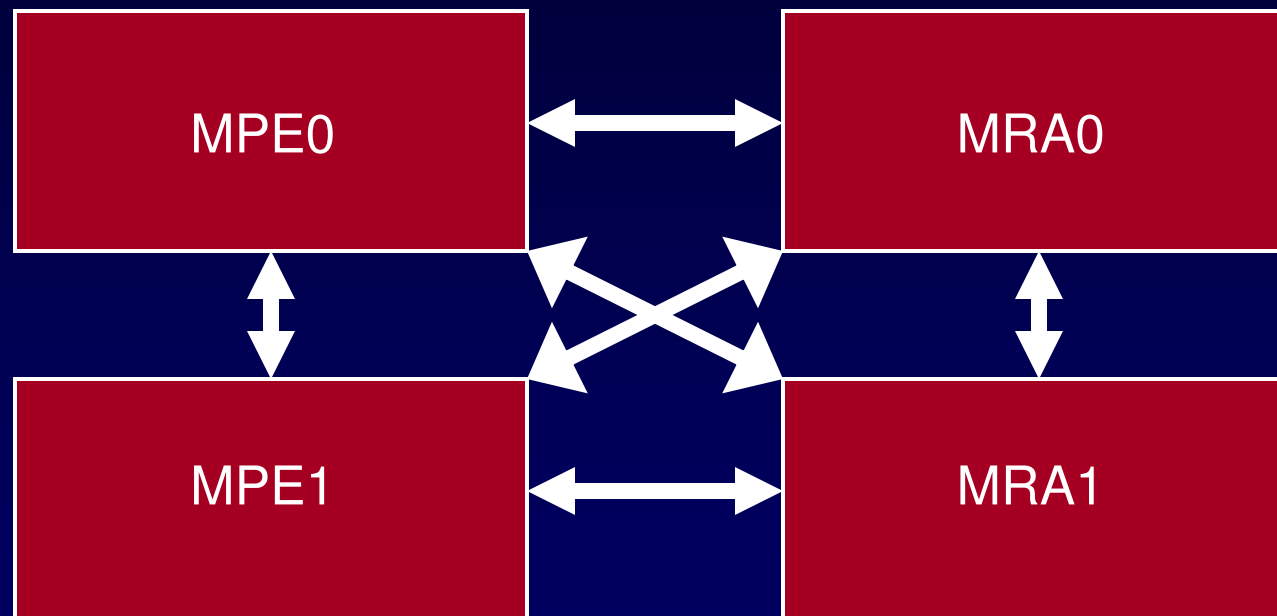
# Case Study #1: Finding a Bug

- In one unit, many assertions failed
  - Mid-transaction address changes
- Needed assumption on inputs

```
module BMpeTop;  
  input IssueTxn, CfgAddress, TxnSent;  
  ...  
  assert_win_unchange ... (...IssueTxn, CfgAddress,  
  TxnSent);  
  ...  
endmodule
```

- *Bug found when assumption fired in simulation*

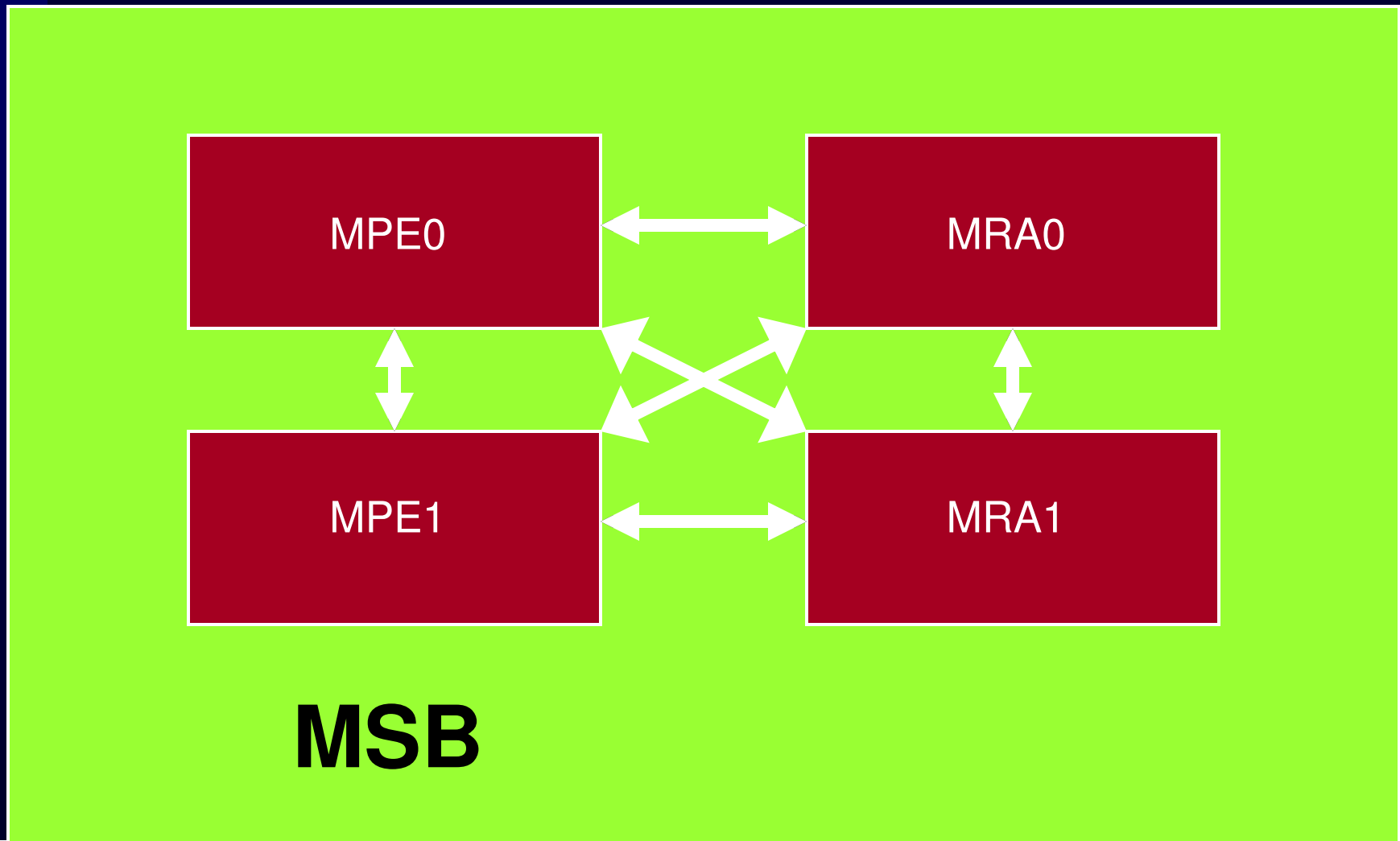
# Case Study #2: Enabling FPV



MPE = Memory Protocol Engine

MRA = Memory Read Arbiter

# Case Study #2: Enabling FPV



# Agenda

- Motivation: Why Formal Property Verification?
- FPV Methodology in Blackford
- Case Studies
- *Results & Recommendations*
- Conclusions, Q&A

# Blackford FPV: Results

- Unit-level FPV
  - DEs found it very useful in design process

# Blackford FPV: Results

- Unit-level FPV
  - DEs found it very useful in design process
- Bugs found
  - 10 interesting bugs found at unit level
    - *Hard to measure since private DE runs not filed*
    - *Many other bugs found as well*
  - 14 FPV-based bugs filed later in project
    - *Simulation always had first shot*

# Blackford FPV: Results

- Unit-level FPV
  - DEs found it very useful in design process
- Bugs found
  - 10 interesting bugs found at unit level
    - *Hard to measure since private DE runs not filed*
    - *Many other bugs found as well*
  - 14 FPV-based bugs filed later in project
    - *Simulation always had first shot*
- Overall Project Confidence
  - 1895 assertions eligible for FPV, 81% proved
  - Excellent logic health
    - *A0 booted targeted OSes within 2 weeks of arrival*

# Recommendations

- Central FPV owner for each project

# Recommendations

- Central FPV owner for each project
- Use same assertions in FPV & simulation

# Recommendations

- Central FPV owner for each project
- Use same assertions in FPV & simulation
- Use FPV in early design stages

# Recommendations

- Central FPV owner for each project
- Use same assertions in FPV & simulation
- Use FPV in early design stages
- Each DE should run FPV on their blocks

# Recommendations

- Central FPV owner for each project
- Use same assertions in FPV & simulation
- Use FPV in early design stages
- Each DE should run FPV on their blocks
- Use assertion density to target FPV gaps
- Integrate FPV in validation testplans

# Open Questions

- How much FPV debugging should we require?
- Resource balance: simulation vs FPV?
- High-level FPV based on architecture?
- Automatically generated assertions?
- Further leverage the synergy between simulation and FPV?

# Agenda

- Motivation: Why Formal Property Verification?
- FPV Methodology in Blackford
- Case Studies
- Results & Recommendations
- *Conclusions, Q&A*

# Conclusions

- FPV is no longer just for PhDs
  - “Turned the usability corner”
  - Successfully utilized by mainstream DEs
- Benefits of FPV for Blackford design
  - Helped DEs think through logic
  - Forced identification of implicit assumptions
  - Found 24+ corner-case bugs
  - Increased overall design confidence

# Q&A



# Backup Slides



# Assertion-Based Verification Maturity Model

1. **Ad-Hoc:** Assertions optionally added by DEs
2. **Planned:** Assertions are expected for simulation
3. **Bug-Hunting:** Some level of FPV is used, bounded proofs OK
4. **Targeted:** FPV run for proofs of high-level intent
5. **Fully Integrated:** FPV is primary unit-level validation method

(model proposed at DAC 2005 by Harry Foster)

# What is OVL?

- Full info available at [www.eda.org/ovl](http://www.eda.org/ovl).

- Usage example:

```
assert_never #(1,0,"problem seen")  
myassert (bclkmain, resetnn, foo!=bar);
```

- More assertion examples:

- Simple boolean conditions: `assert_never`
- Bit vector checking:  
`assert_{one,zero_one}_{hot,cold}`
- Check behavior between events:  
`assert_win_change`
- Sequential behavior: `assert_next`, `assert_time`,
- Track signal value changes: `assert_delta`
- More complex behaviors: `assert_fifo_index`