Abstract

NI-SAR (NASA ISRO Synthetic Aperture RADAR) is designed to observe and take measurements of some of the planet's most complex processes, including ecosystem disturbances, ice-sheet collapse, and natural hazards such as earthquakes, tsunamis, volcanoes and landslides[1].

NI-SAR utilizes 21 FPGAs; 8 unique designs to perform its complex radar processing and control. A 2 year verification effort was allotted to verify all FPGA designs.

UVM was chosen to be the primary verification vehicle. The aggressive schedule however makes the UVM ramp-up period an effective downtime period per unit verification.

Formal methods are applied during the UVM ramp-up period to provide an earlier and constant verification effectiveness.

[1] https://nisar.jpl.nasa.gov/nisarmission/

Maximizing Verification Quality & Resources with Formal and UVM

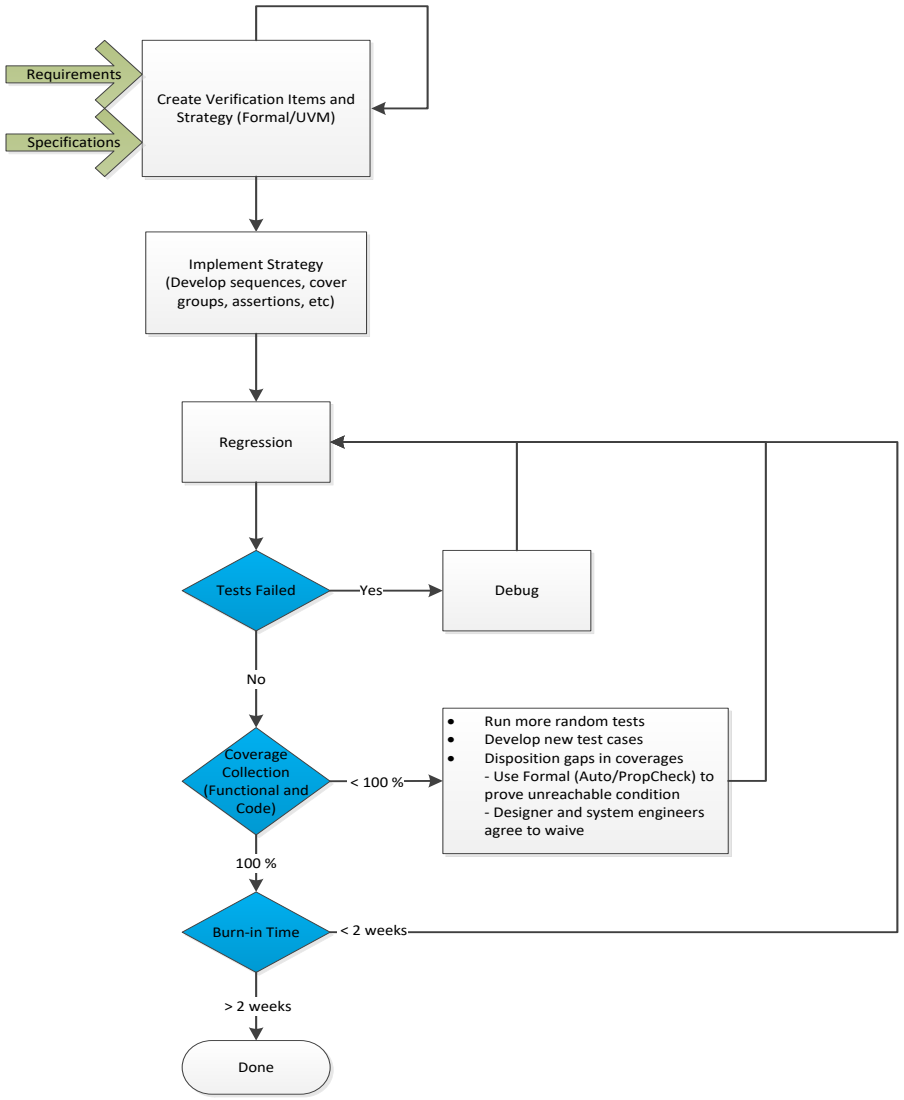
Chester Lim – Jet Propulsion Laboratory, California Institute of Technology

JPL Jet Propulsion Laboratory
California Institute of Technology

Motivation

- Previous projects utilized FPGAs (< 5) with small designs in them. Design and verification were viewed as the same effort and performed by the same engineer. A typical schedule for FPGA development was roughly 12 to 18 months
  - SMAP (Soil Moisture Active Passive)(2015) used 4 FPGAs – 2x RTAX1000, 2x QVirtex-2
- NI-SAR (2022) mission uses 21 FPGAs – 3x RTAX2000, 6x RTProASIC3000E, 12x QVirtex-5
  - 8 unique designs – roughly > 65% utilization
  - Independent verification team utilized, 2 years to verify all designs – roughly 3 months per unique design
- UVM is the primary verification vehicle however the effective ramp-up takes roughly 1.5 months (half the allotted schedule!!)
- Formal methods are applied during the UVM effective ramp-up period to provide an earlier and constant verification effectiveness

NISAR's Verification Process



- Verification begins with a specification walkthrough with the designer and system engineers
  - Specs and requirements are flushed out and verification items are created
  - Each verification item is assigned a strategy: UVM covergroups, UVM assertions, Formal assertions, or hardware tests
- Implementing verification strategy for UVM takes roughly 1.5 months to go from spec/requirement walkthrough to adding first covergroup/assertion
- Implementing verification strategy for Formal happens almost immediately even prior to completion of spec/requirement walkthrough
- Regression starts when at least one UVM test is passing
- Functional coverage is collected continuously. Code coverage is collected when there are no more tests and covergroups/assertions to add.
  - Gaps in coverages are looked at and addressed accordingly
    - Run more random tests
    - Develop new test cases
    - Waive the gap based on Formal check (unreachable/uncoverable) or inputs from designer and system engineers.
- Verification is complete when regression runs successfully for more than 2 weeks with 100% coverage + waivers

Formal/UVM Effectiveness

Formal

Pro

- Perform structural checks and essential sequential checks on design even before completing spec/requirement walkthrough
  - Clock & Reset Domain Crossing (CDC & RDC)
  - Linting & Code clean up (AutoCheck)
- Does not require a testbench!!
  - "Shallow" to "moderate" properties such as simple state transitions can be written concurrent to formulation of verification items (PropCheck)

Con

- Due to the exhaustive nature of Formal, "deep" properties such as complex state transitions have a difficult time converging in a reasonable amount of time

UVM

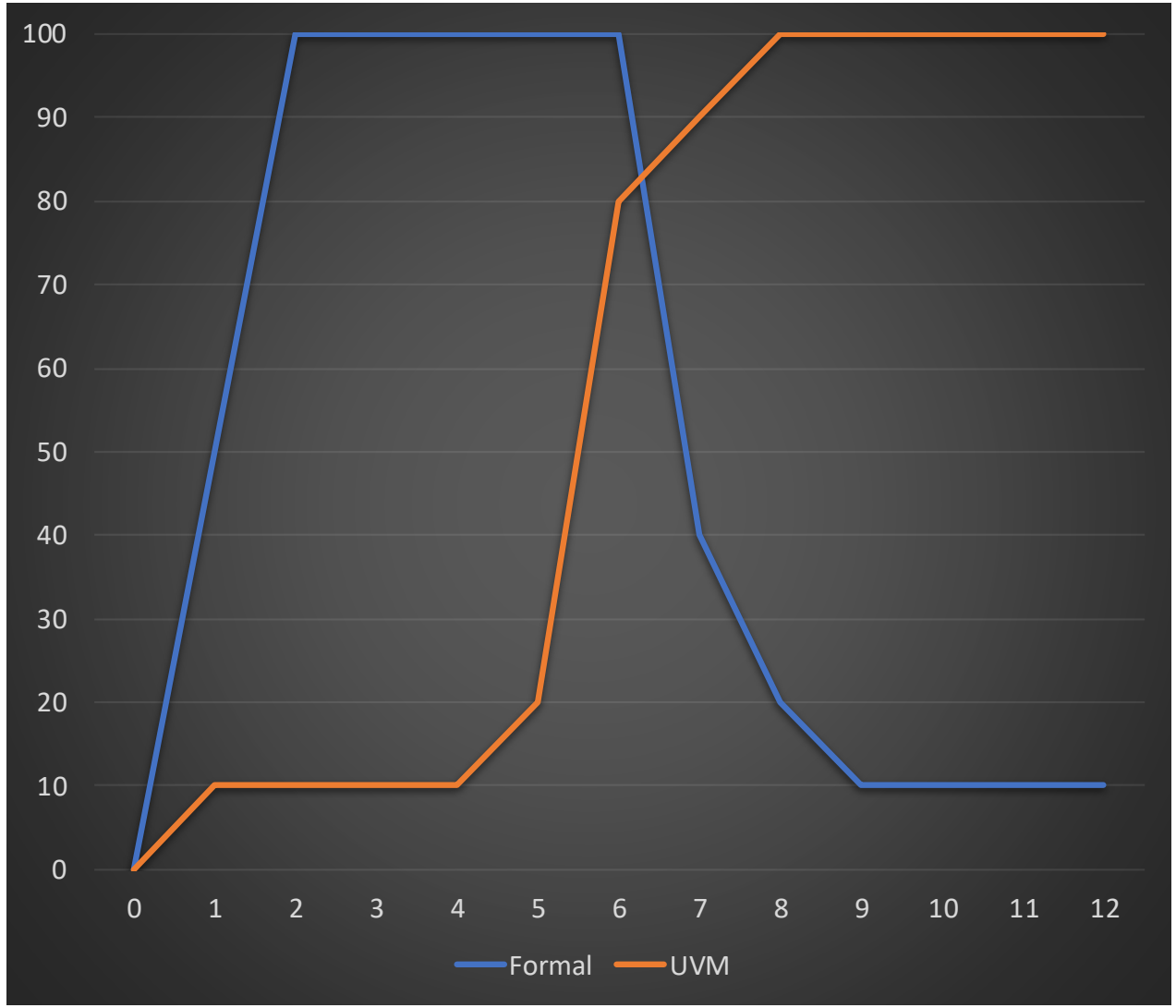
Pro

- Able to reach deep state spaces more readily compared to Formal by tailoring the test sequences
- Provides automated tracking of functional and code coverages

Con

- Requires a ramp-up period to build the UVM framework and testbenches. Begins after completing spec/requirement walkthrough
- Non-exhaustive and can leave potentially untested corner cases
  - Relies on volume regression runs due to random nature of stimulus. Can only provide a statistical odd versus a proof (Formal)

Verification Timeline



Overall process

- **Week 1-3**: Spec/requirement walkthrough and majority of the VI formulation occurs.
  - VI formulation can happen throughout the process as more information is uncovered that was not reviewed during the initial spec/requirement walkthrough
- **Week 11-12**: Regression burn-in

Formal

- **Week 1-2**: Run AutoCheck within the 1st week and CDC/RDC within the 2nd week after port domain information is gathered from spec/requirement walkthrough
- **Week 1-8**: Designers work on clearing CDC, RDC, and AutoCheck issues
- **Week 3-5**: Properties are formulated based on VI (with a Formal strategy). Work on converging ”shallow” to ”moderate” properties.
- **Week 5-9**: Work on converging “deep” properties
- **Week 9-10**: Using a combination of AutoCheck and PropCheck to aid with code coverage closure

UVM

- **Week 1-4**: Gathering specs/requirements & formulating VI
- **Week 5-6**: UVM infrastructure is created
- **Week 6-9**: Adding sequences, predictors, and covergroups/assertions
- **Week 9-10**: Coverage is collected and addressed

Summary

- UVM is the primary verification vehicle. An aggressive schedule makes the ramp-up period critical (low verification effectiveness)
- Formal is applied during the UVM ramp up to provide an earlier and constant verification effectiveness.
  - CDC, RDC, and AutoCheck is used to verify structural soundless of the design even prior to completion of spec/requirement walkthrough
  - PropCheck is used to verify “shallow” to ”moderate” (i.e., simple state transitions) specs/requirements without needing to develop a testbench
- Use AutoCheck and PropCheck during coverage closure process to determine the reachability of a portion of the design.
- Formal and UVM are complementary and when used effectively, provides an earlier and constant verification effectiveness.