2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Accelerating Error Handling Verification Of Complex Systems: A Formal Approach

Bhushan Parikh, Peter Graniello, Neha Rajendra

Intel Corporation

intel

accellera SYSTEMS INITIATIVE

Agenda

- Acronyms
- Introduction
- Proposed Methodology
- Results
- Summary

Acronyms

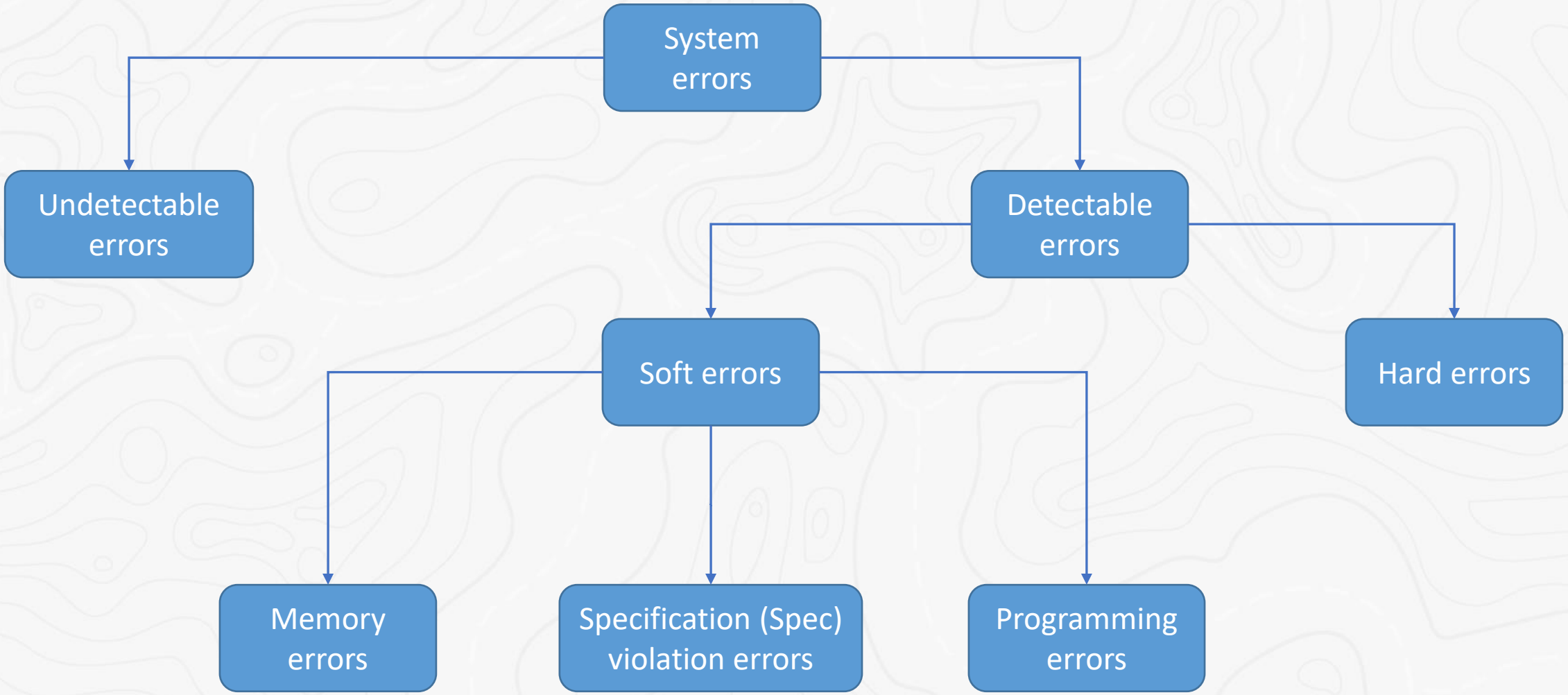
- CBRV => Constraint Based Random Verification
- FPV => Formal Property Verification
- CEX => Counter Example (/Failure)
- FSM => Finite State Machine

Error Handling Verification: Vital & Challenging

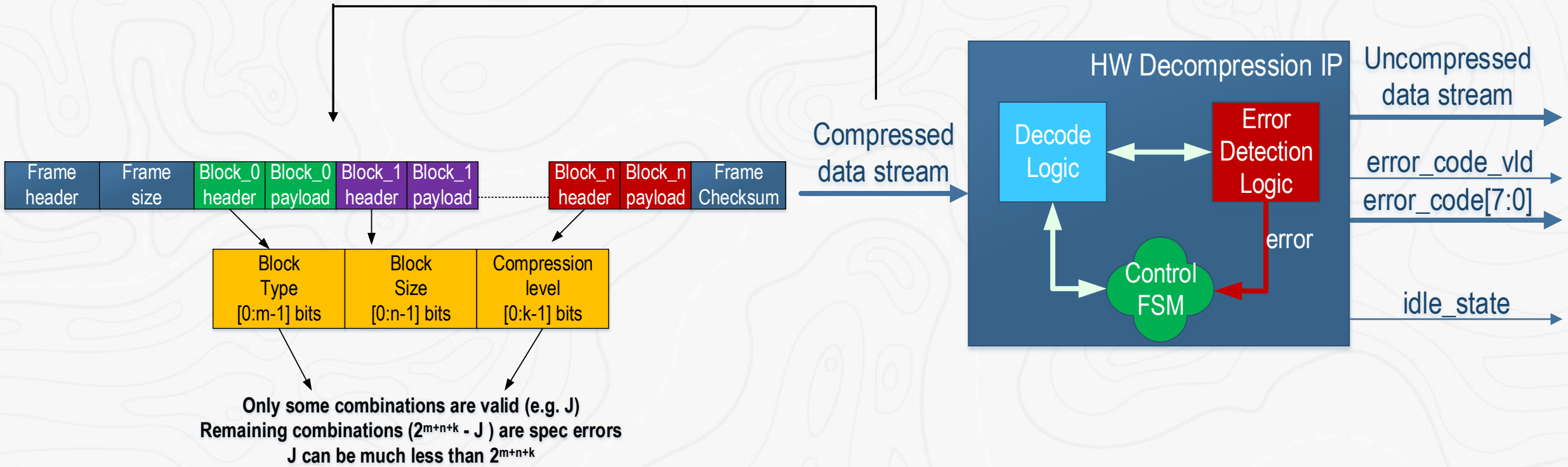
- Vital
  - One of the key components in grading the reliability of a system

- Challenging
  - Computer systems have advanced significantly and are now distributed systems
  - Vast number of possibilities that can result in an error due to increased complexity

Classifications Of Errors



Example Of A Complex System



Requirements & High-level Verification Plan

- Error handling requirements
  - All spec errors shall be detected correctly
  - Detectable errors shall be reported with the respective error code
  - Shall gracefully complete the processing

- High level Verification Plan
  - Error detection verification
    - Verify that detectable errors are detected correctly
  - Error reporting verification
    - Verify that detected errors reported with correct error code
  - Graceful completion verification
    - Verify that the system gracefully completed processing of any input data

Challenges Associated With CBRV Methodology

- Difficult to generate test vectors with spec error
  - SW compressors are spec compliant and complex to modify
- Too many combinations to verify
  - Assertion of an error is an asynchronous event with respect to the rest of the design functionality
- Manual and time consuming
  - Identification and development of directed test-cases required for completeness

How About FPV?

- Widely accepted methodology and mainly used to accomplish normal (i.e., non-error) functional verification goals

- However, can be very effective achieving error handling verification related goals

Primary Inputs Are Free Nets In FPV

- Formal engines will exercise every possible combination of the required stimulus
- Need to specify only the expected behavior for errors

Header Multibit Fields		
Block Type	Compression Level	
Type A	1-20	Valid
	0,21-31	Spec Violation
Type B	1-9	Valid
	0,10-31	Spec Violation

- CEX due to under-constrained formal environment most likely translate to actual bugs

Error Detection Verification (CBRV vs FPV)

- E.g., verify that compressed stream violating block size specification is detected correctly (i.e., BLOCK_SIZE <= SPEC_BLOCK_SIZE)
  - BLOCK_MAX_SIZE = Maximum possible value of the field
  - SPEC_BLOCK_SIZE = Maximum value supported by the specification
  - BLOCK_SIZE = Decoded value from the compressed stream
- CBRV
  - # of test cases = BLOCK_MAX_SIZE – SPEC_BLOCK_SIZE
- FPV
  - Only two assertions are required,
    - (BLOCK_SIZE > SPEC_BLOCK_SIZE) |-> ##[0:$] block_size_error
    - block_size_error|-> (BLOCK_SIZE > SPEC_BLOCK_SIZE))
  - FPV assertions and methodology can be applied to all specification errors

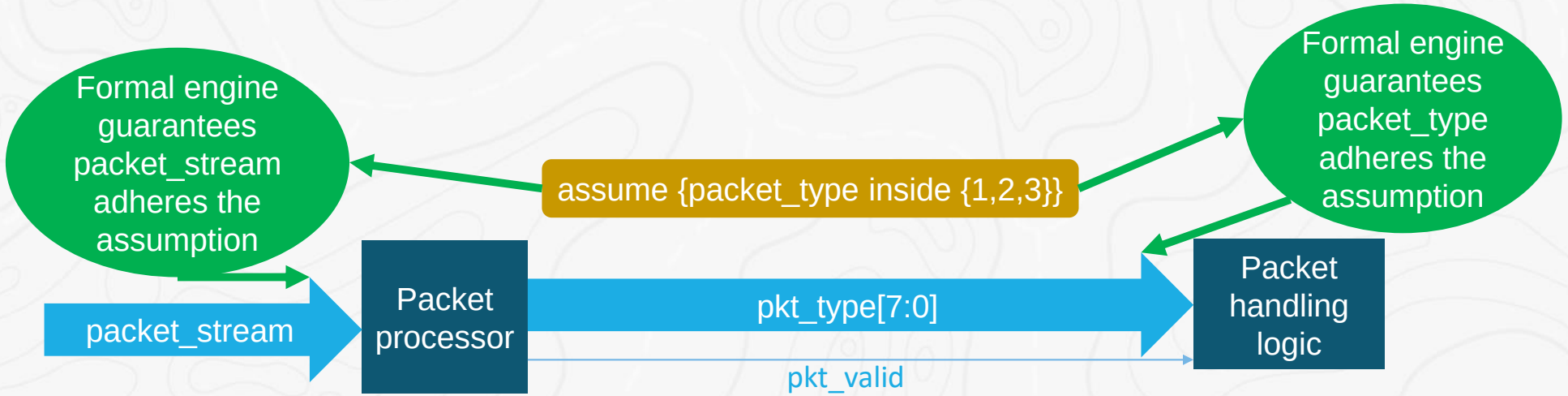
Lossless HW Decompression IP Spec Errors

- Other examples of specification errors for the lossless HW Decompression IP

Spec Error	Explanation
Unbalanced Huffman Tree	The Huffman Tree used for compression is not balanced
Received incorrect code	The Huffman code does not match with encoding of any symbol from the compressed stream
Distance out of range	The copy location in token is too far behind
Incomplete Stream	The compressed stream is not complete
Padding Byte Error	Padding bytes in the compressed stream are corrupted

Constraints Can Back-propagate In FPV

- Exercise complex combination of stimulus at top level using simple assumptions at lower level



Error Reporting Verification (CBRV vs FPV)

- E.g., verify that detected error condition is reported using correct error code
  - ERROR_CONDITION_N = Detected error condition
  - ERROR_CODE_N = Expected error code for ERROR_CONDITION_N

- CBRV
  - Minimum # of test cases = # of supported error codes ($(2^8 – 1)$ = 255 in our case)

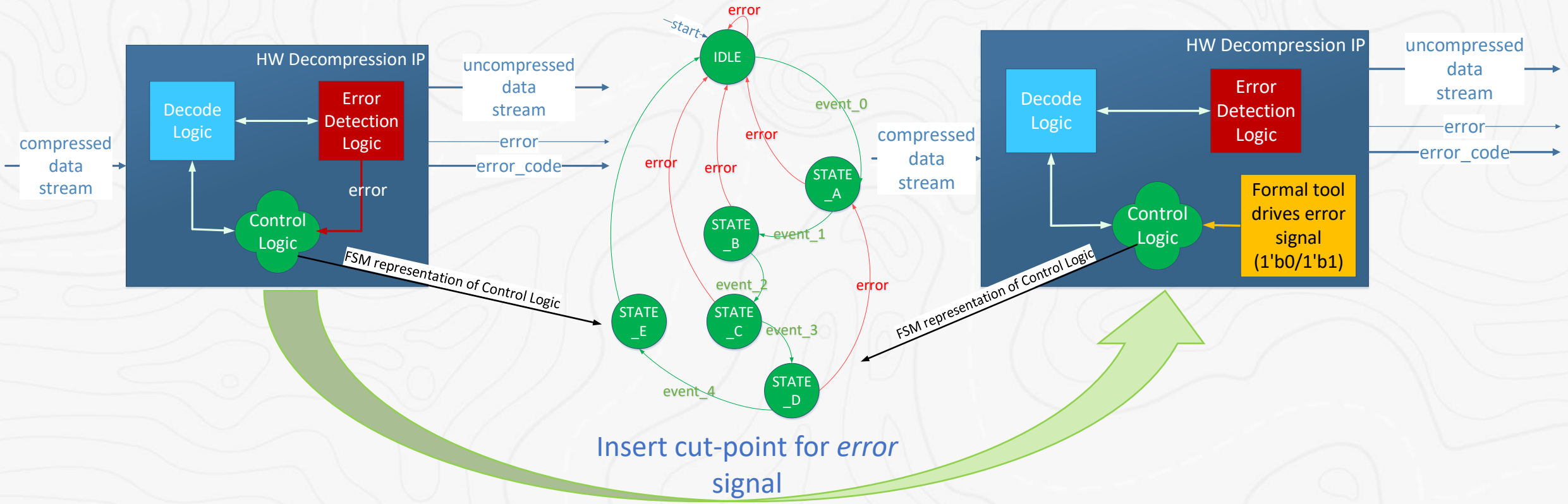
- FPV
  - Only two assertions are required,
    - (error_condition == ERROR_CONDITION_N) |-> ##[0:$] ((error_code == ERROR_CODE_N) && (error_code_vld))
    - ((error_code == ERROR_CODE_N) && (error_code_vld)) |-> (error_condition == ERROR_CONDITION_N)

Graceful Completion Requirement

- What is it?
  - After processing any request, system shall gracefully exit to a known (or an IDLE) state and be serviceable
- Importance
  - A key reliability aspect especially for network accelerators (e.g., lossless HW Decompression IP)
  - Any bug in the implementation may lead to Denial of Service (DoS) attacks
    - E.g., an attacker with an erroneous compressed stream can put the entire system in an unrecoverable state and making it unavailable for rest of the users

FPV Allows Cut-point Insertion In Design

- Can convert any signal in the design to a free net



Graceful Completion Verification (CBRV vs FPV)

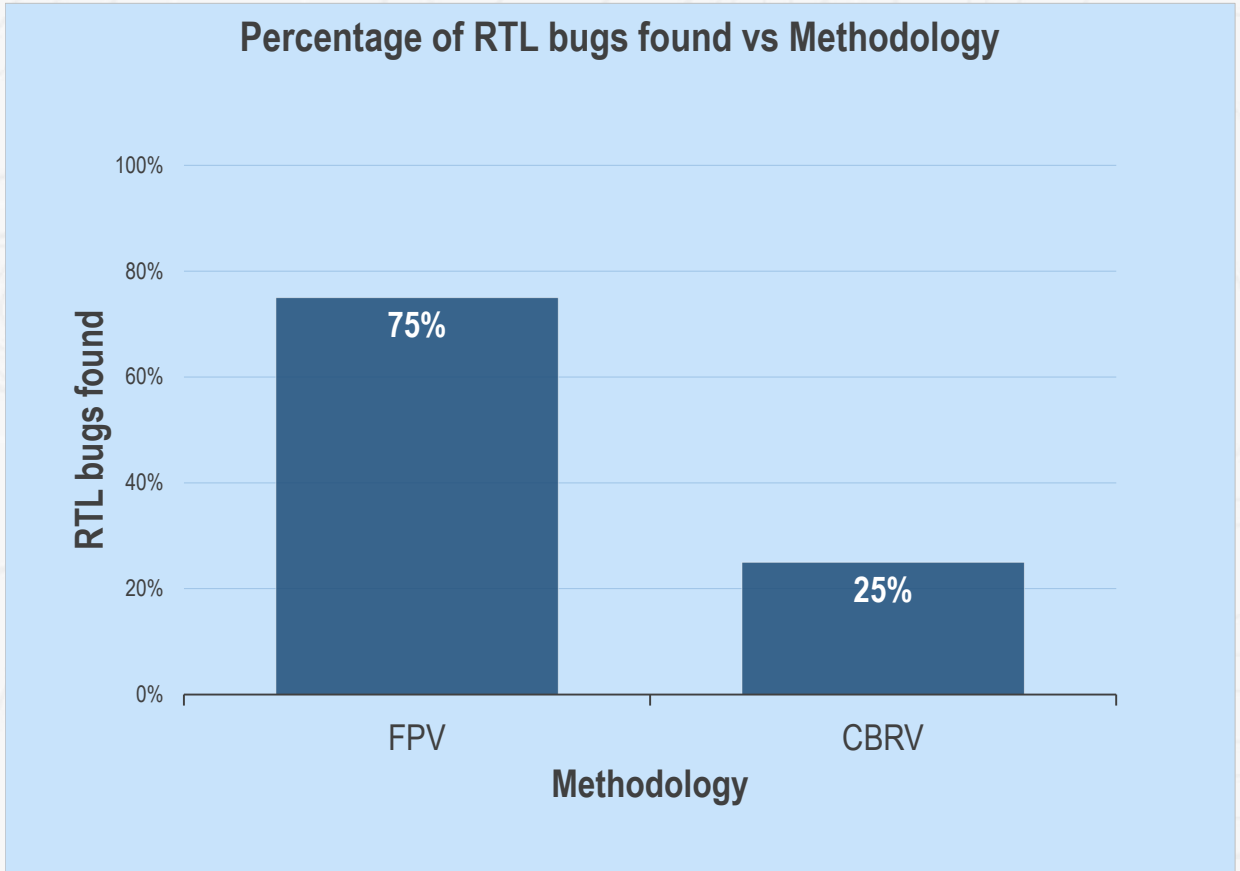
- E.g., verify that control FSM reaches to IDLE state for any error condition

- CBRV
  - Develop a test to assert the *error* signal
  - Modify the test to ensure that error signal is asserted for every state
  - Develop and analyze functional coverage to verify that all possible cases are covered

- FPV
  - Create cut point for the *error* signal
  - Only two assertions are required,
    - $rose(error) |-> ##[0:$] idle_state
    - (error && idle_state) |-> ##1 idle_state

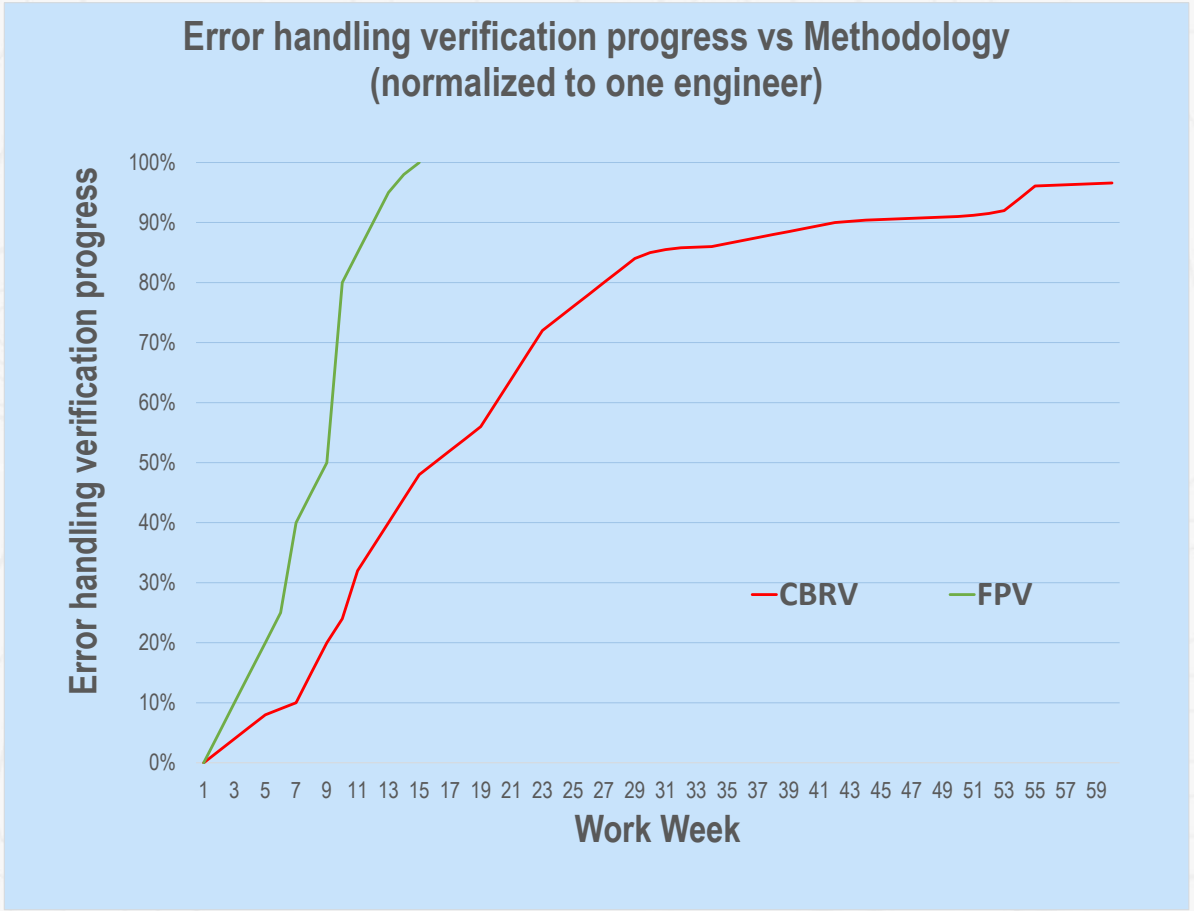
Results (1/2)

- Despite its late deployment, FPV found 75% of the overall error handling verification bugs
  - Bugs were complex and could not have found using CBRV
  - For completeness, reproduced 25% of the remaining bugs using FPV



Results (2/2)

- Completed error handling verification task well ahead of planned schedule
  - Leveraged various industry standard techniques and achieved 100% convergence for all assertions
  - Various formal coverage metrics and iterative reviews of FPV test-plan to sign-off



Summary

- Discussed error handling verification and challenges associated with achieving sign-off for it using CBRV methodology
- Demonstrated the effectiveness of applying FPV methodology to address these challenges
- Future work includes leveraging FPV methodology in security verification

Thank you!

Questions?