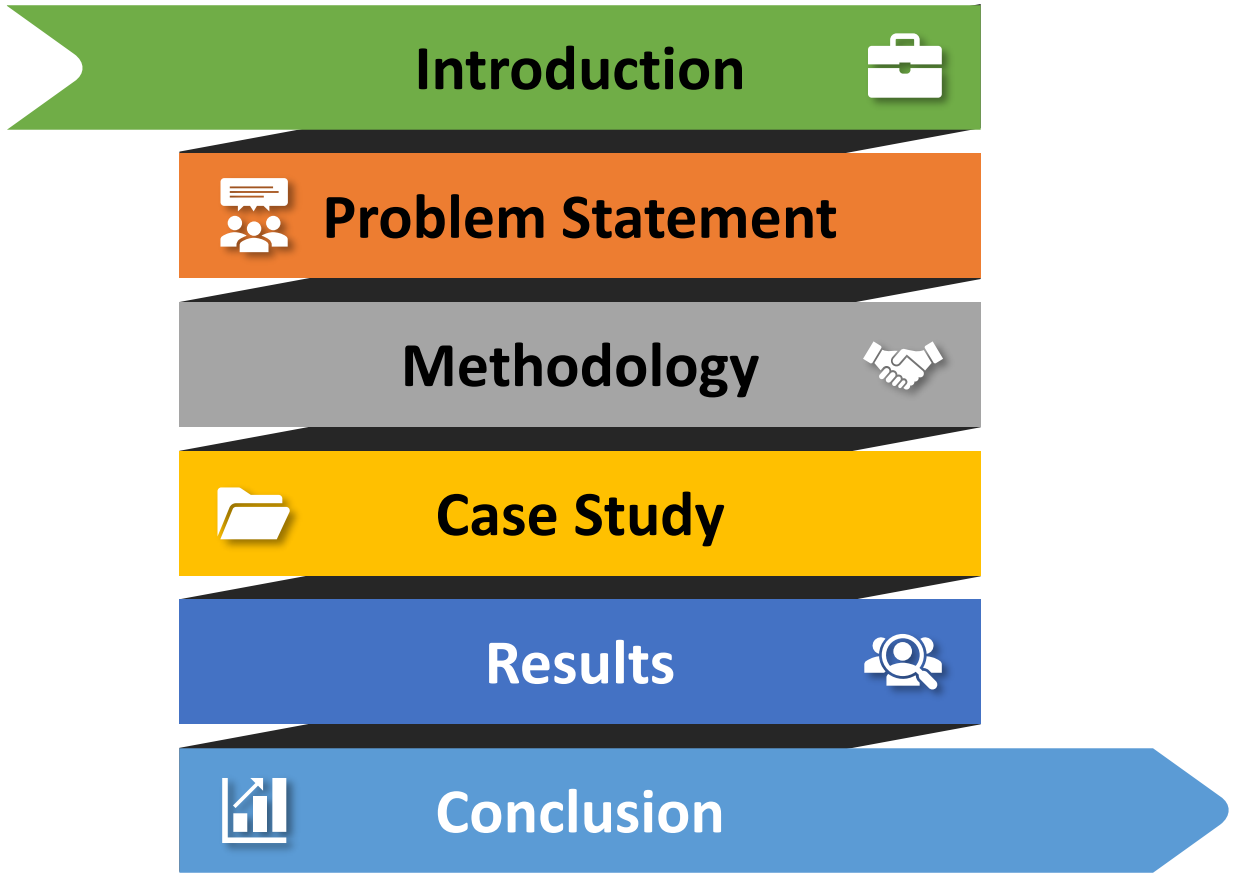
2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Formal Verification Approach to Verifying Stream Decoders: Methodology & Findings

Abhishek Asi, Anshul Jain, Aarti Gupta

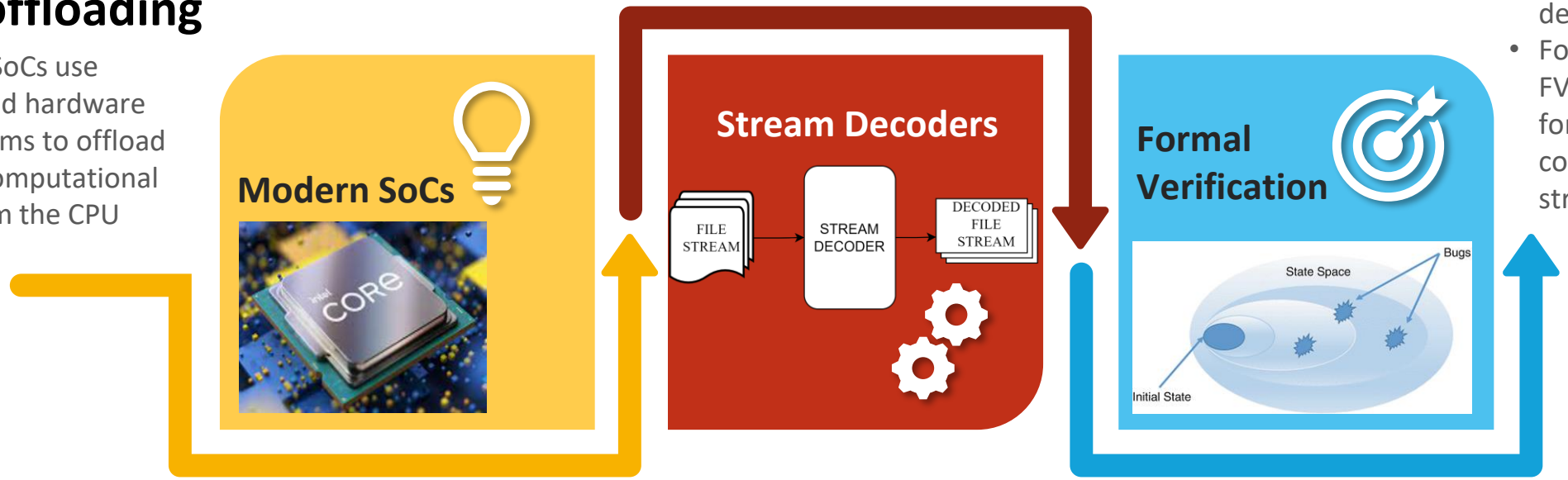
Agenda


Introduction
Problem Statement
Methodology
Case Study
Results
Conclusion

Introduction

CPU - offloading

Modern SoCs use specialized hardware sub-systems to offload certain computational tasks from the CPU



Aim of the presentation

- Explore FV approach tailored for stream decoders
- Foster confidence in FV as a robust tool for ensuring the correctness of stream decoders

Problem Statement

Importance of Stream Decoders

Correct functionality crucial to prevent data corruption or loss

Rising complexity

Due to evolving formats and standards

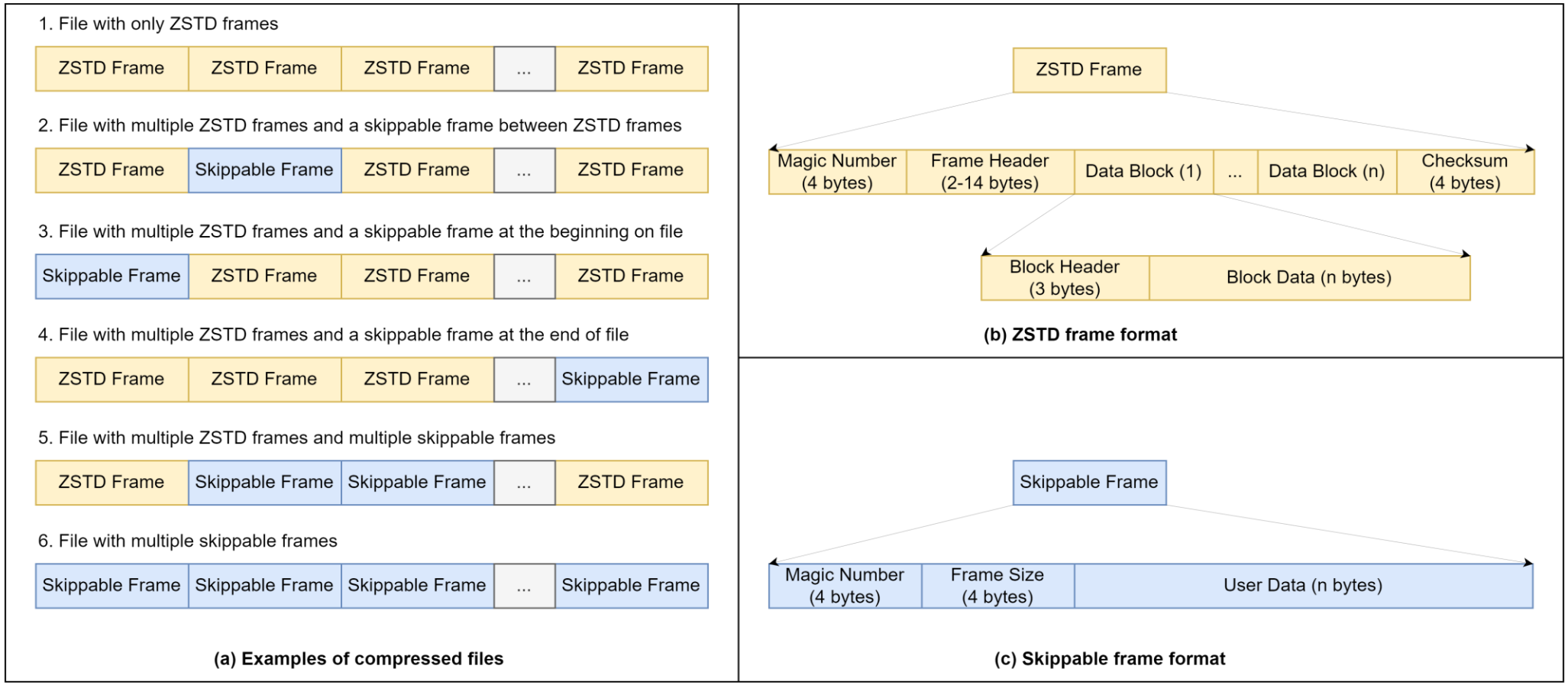
Limitations of conventional methods

Conventional methods miss corner-case scenarios or subtle issues that only a specific file can cause

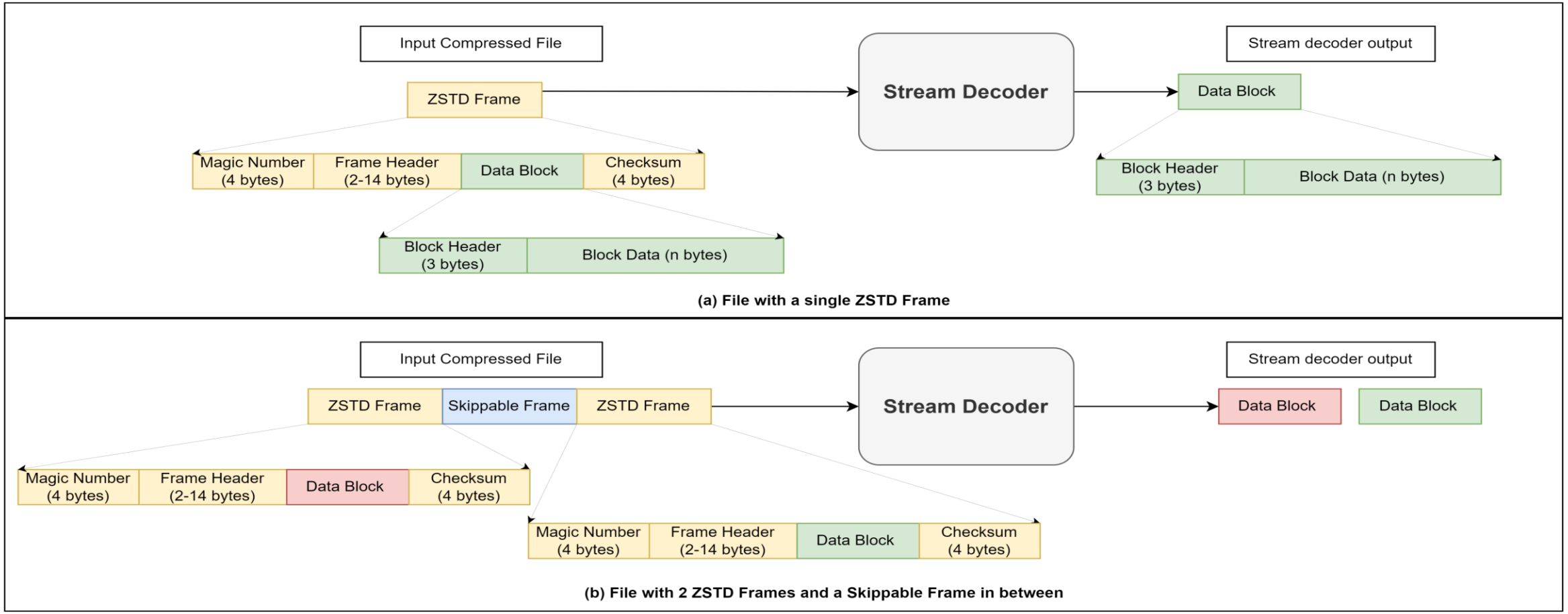
Power of Formal verification

FV provides comprehensive validation and mathematically confirms design accuracy

ZSTD file format

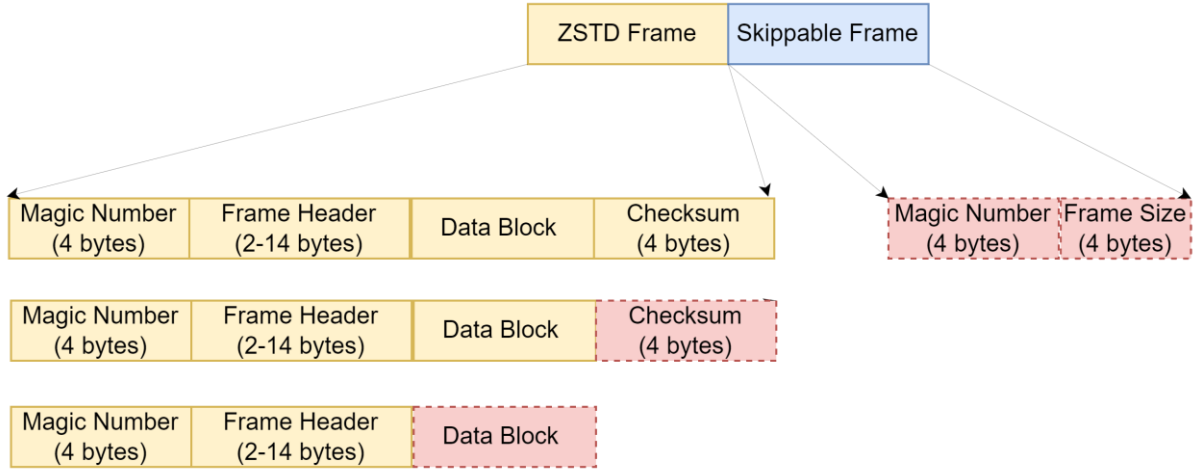


Basic functionality of stream decoders



Complexity of ZSTD stream decoders

Variable	Sub-variable	Range
Frame Header	Frame Header Descriptor	4 flags, one each for – frame content size, single segment, checksum, and dictionary
	Window Descriptor	0 to 2^{64} - 1 bytes (16 Exabytes)
	Dictionary ID	4 bytes can represent an ID 0-4294967295
	Frame Content Size	0 - 2^{64} – 1
Data Blocks	Number of Blocks	1- ∞
	Block Size	0 - 2^{21} – 1
	Block Content	Arbitrary
Content Checksum	Optional	Arbitrary
Skippable Frames	User Data	Arbitrary
File Structure	Number of Frames	1- ∞
	Types of Frames	ZSTD only, skippable only, both
	Frame Order	Different arrangements of ZSTD and skippable



Examples of corrupt input compressed files.

[Refer to the paper for more examples]

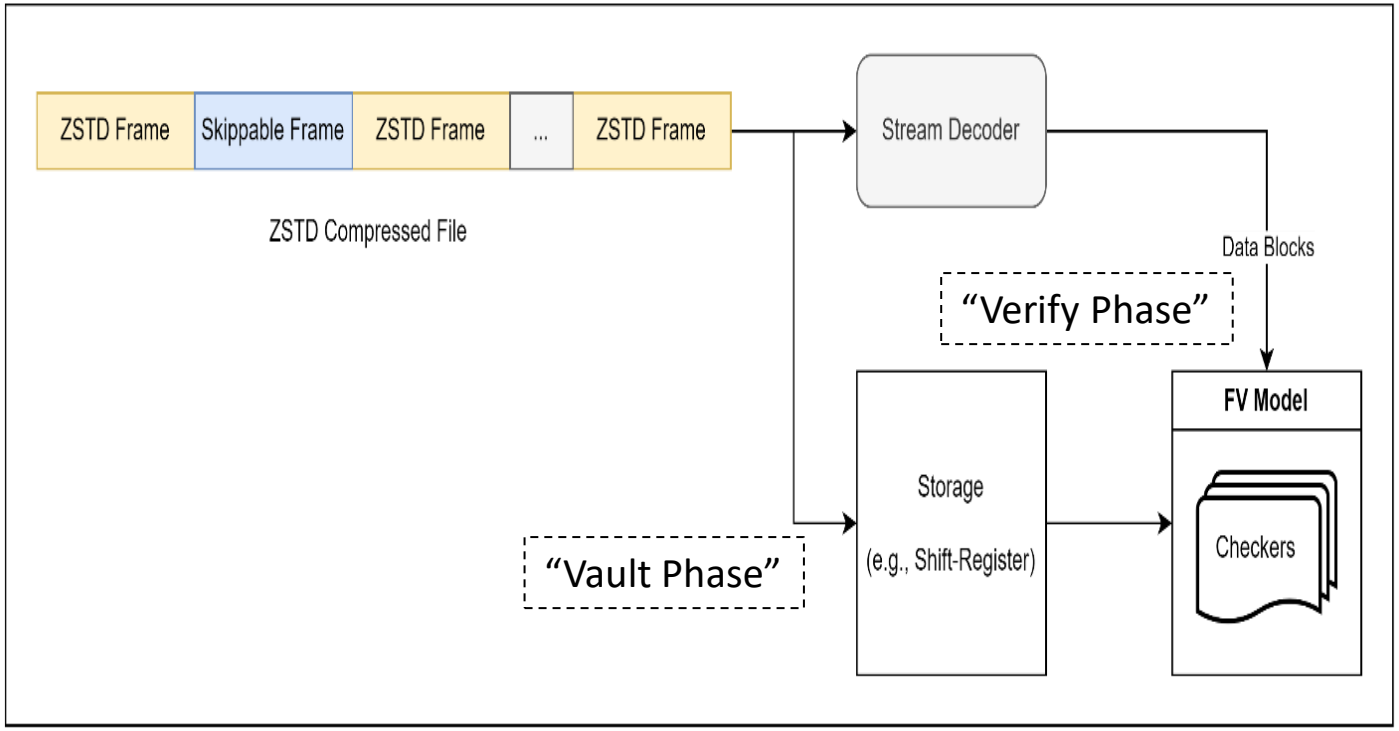
Methodology Solution #1: Vault & Verify

Complexity in handling compressed files

Coverage Scalability

Incapable of Detecting Hangs

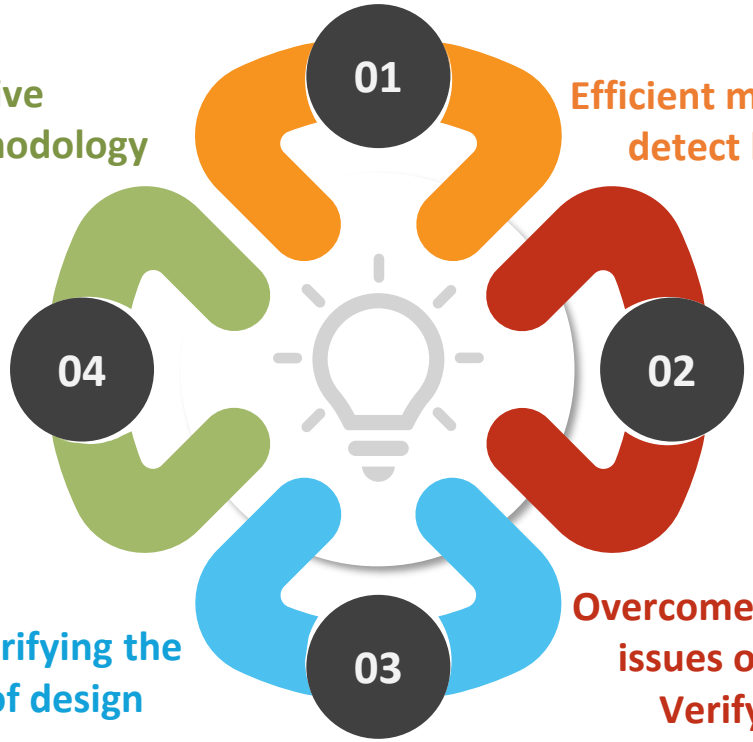
Challenges with Vault & Verify



Methodology Solution #2: Quiesce Check

In-exhaustive verification methodology

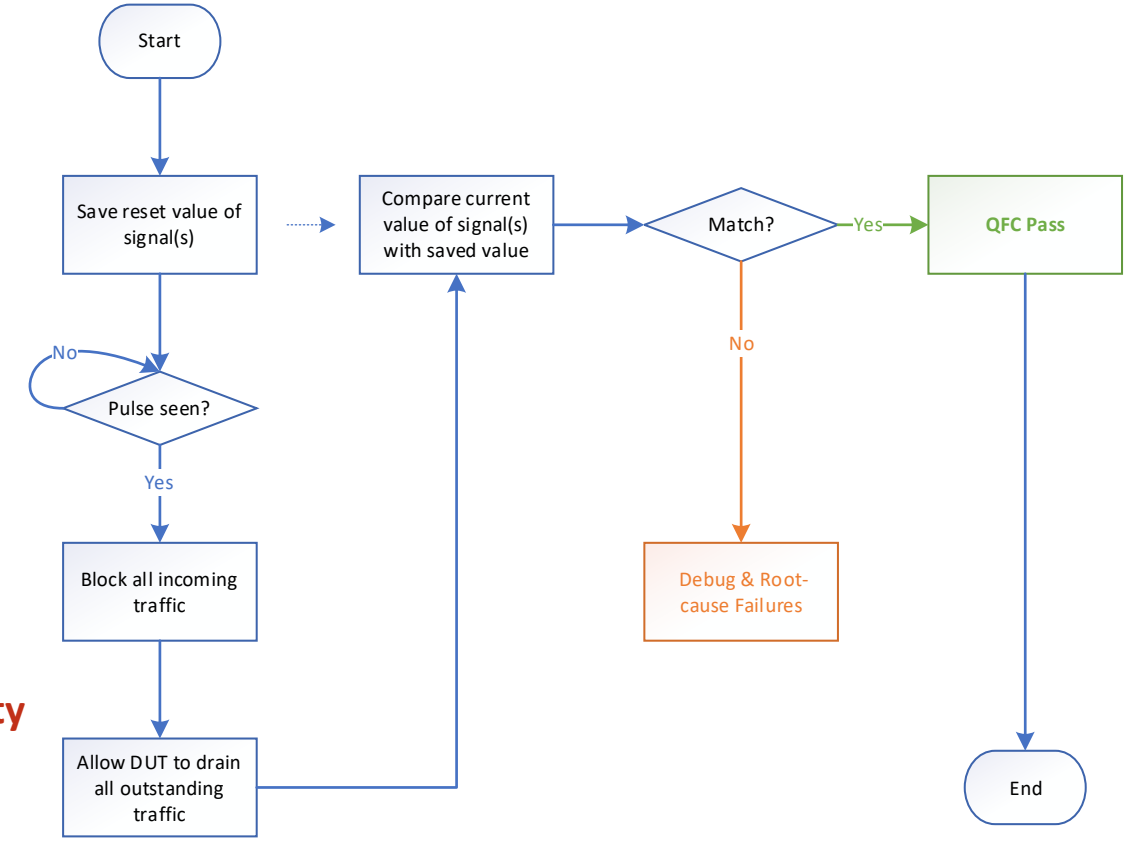
Efficient method to detect hangs



Incapable of verifying the correctness of design behavior

Overcomes the scalability issues of the Vault & Verify approach

Challenges with Quiesce Check



Quiesce Formal Checking Framework

Methodology Solution #3: Instant Inspection

Dissection of Verification Goals into five major areas

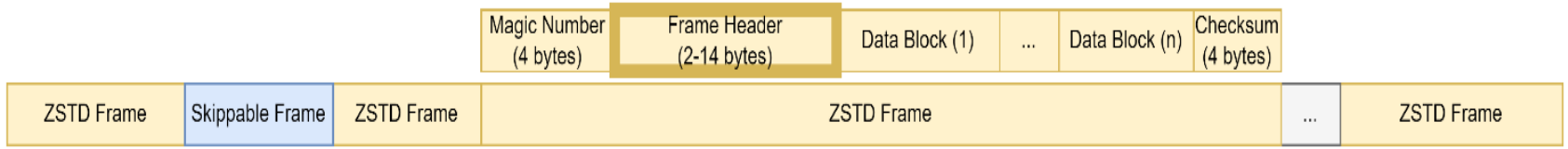
- Status Flag Verification
- Data Integrity
- Data Ordering
- Latency Verification
- Error Handling

Sliding Window Analysis

Instead of analyzing the entire compressed file, a smaller sliding window of the file is analyzed

ZSTD Compressed File

The sliding window width should be 14 bytes, which is the largest chunk of control information available in contiguous form



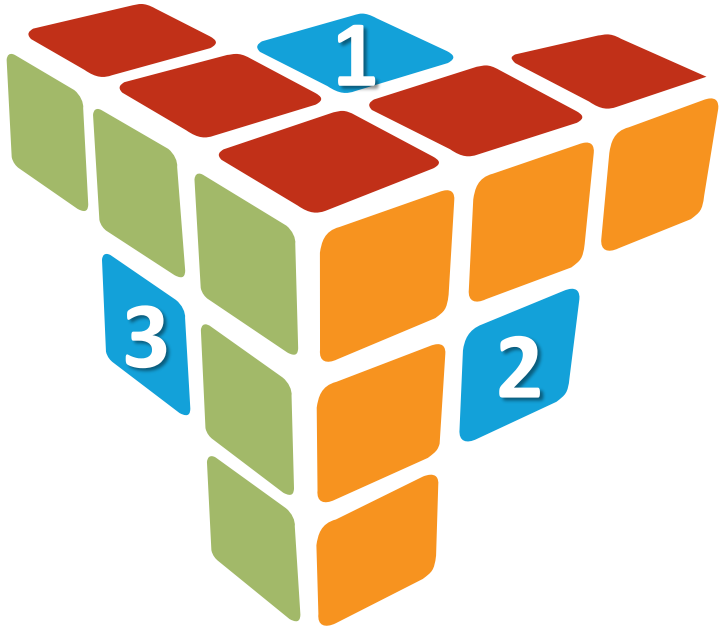
Challenges with Instant Inspection

Requirement of Detailed Description

Detailed descriptions of intermediate checkpoints, typically documented in microarchitectural specification

Gaps in Specifications

Considering the vast amount of data a decoder processes, ensuring the microarchitectural specification is complete and unambiguous is difficult



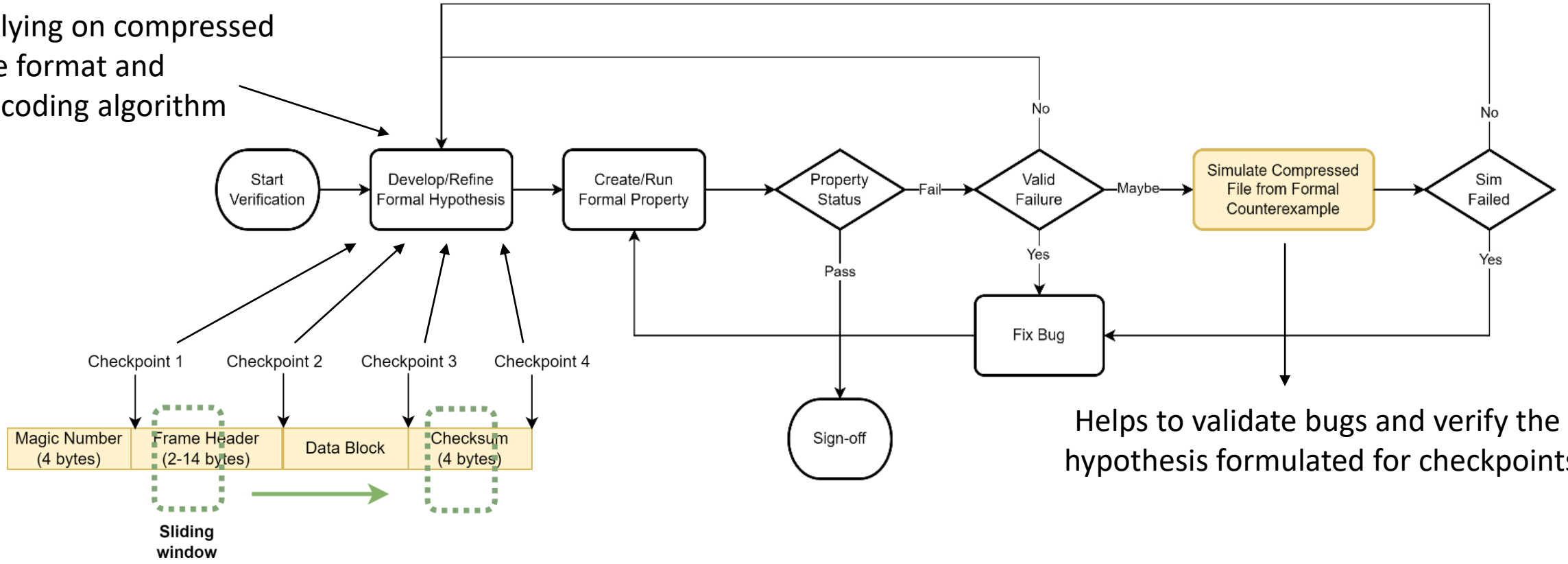
Complexity in Creating Specifications

Crafting comprehensive microarchitecture specification that accounts for all possible scenarios is challenging

Hypothesis based Property Verification

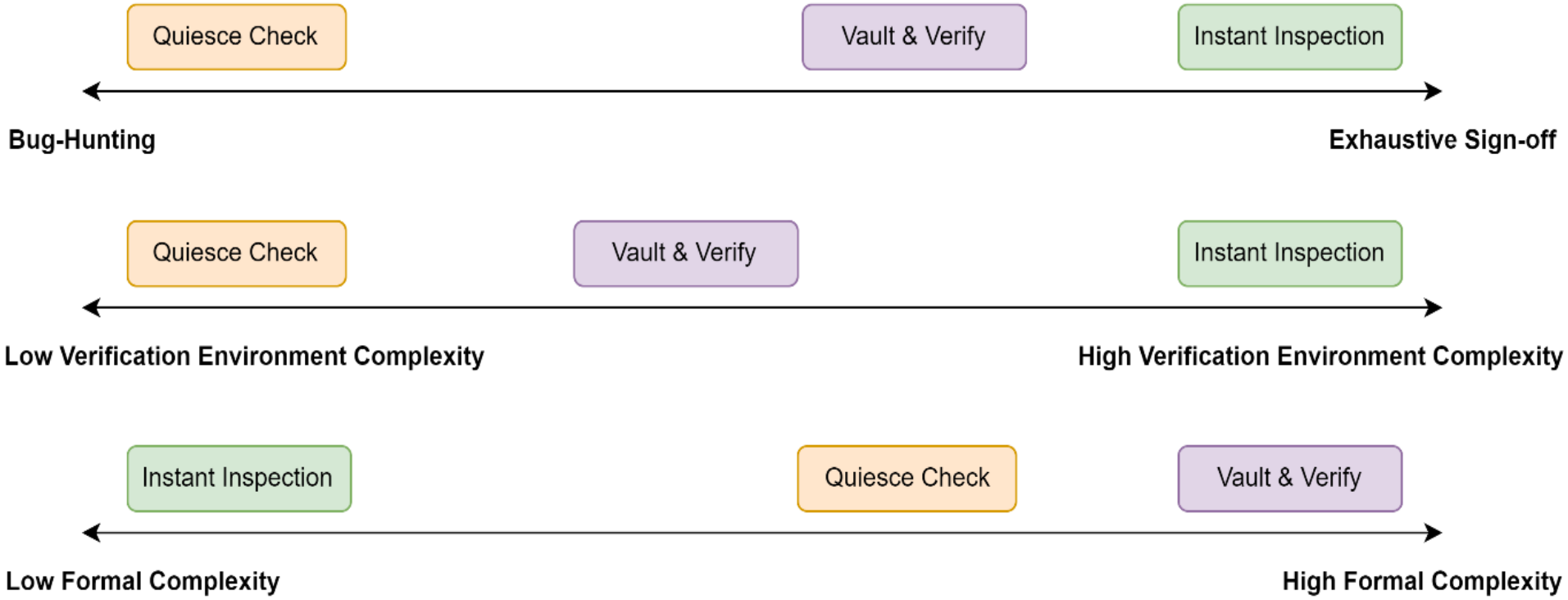
Hypothesis-based method tackles the challenges of Instant Inspection

Relying on compressed file format and decoding algorithm



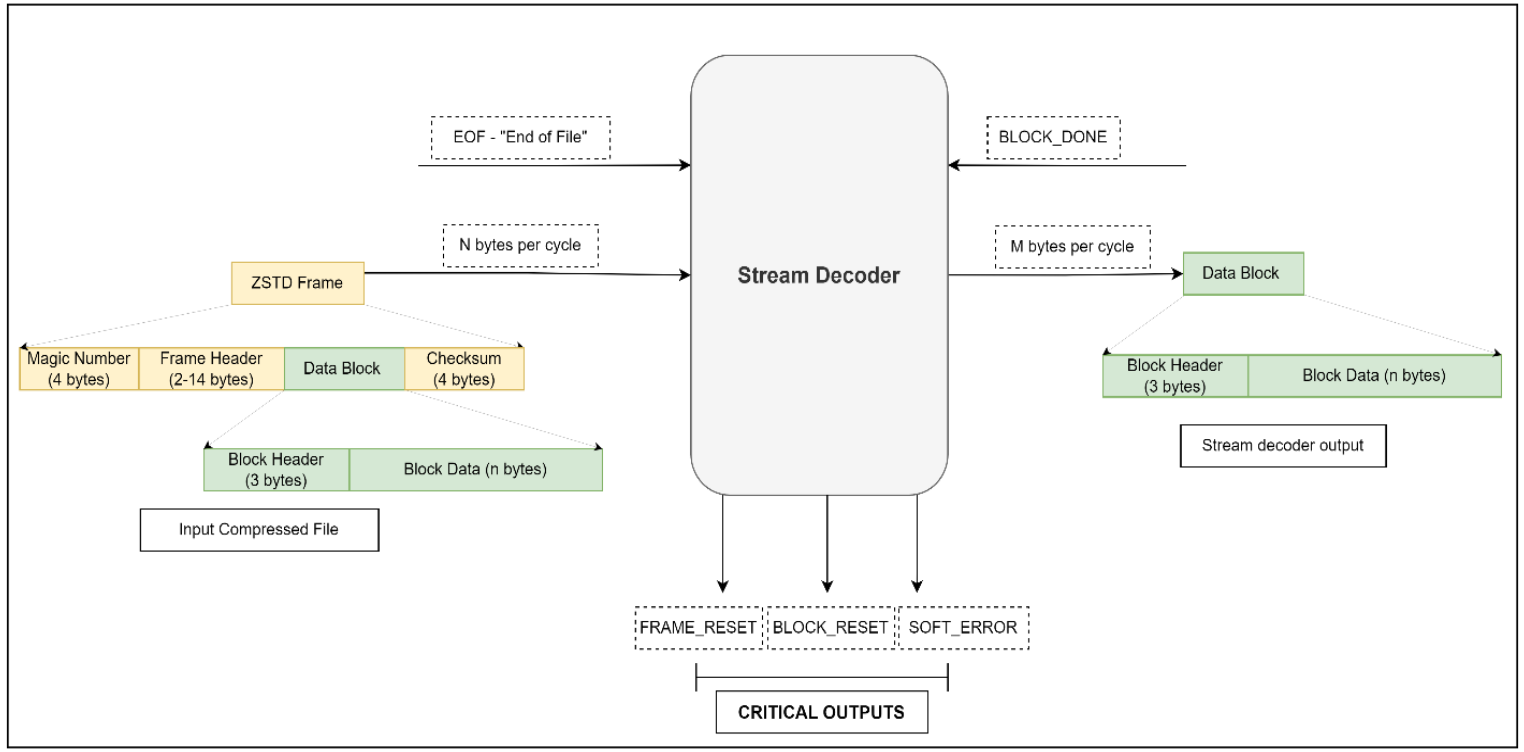
Helps to validate bugs and verify the hypothesis formulated for checkpoints

Comparative view of suggested solutions



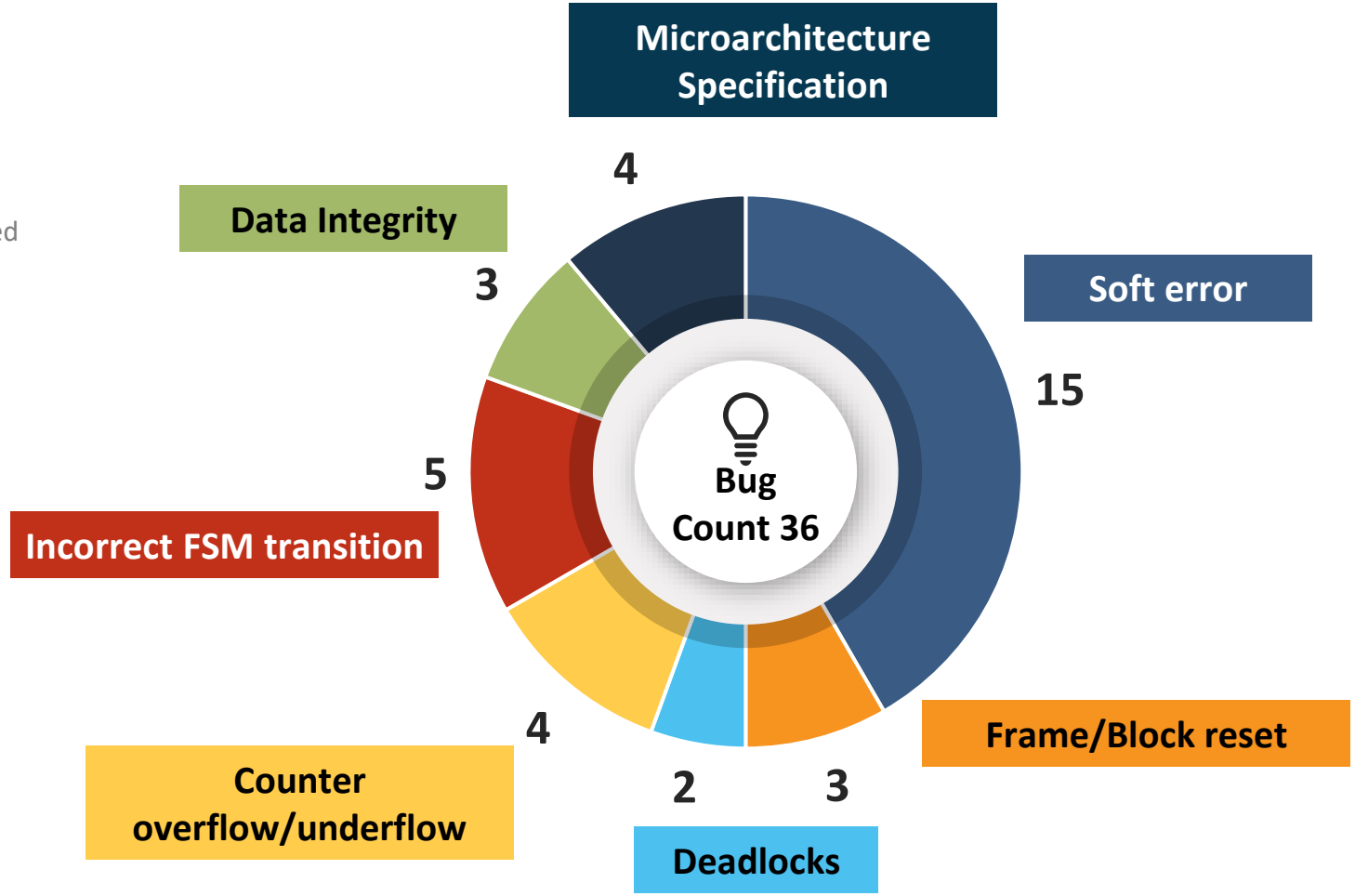
Case study: ZSTD Decompression Stream Decoder

ZSTD Stream Decoder used in Intel's Xeon CPUs targeted for data-center applications. Refer paper for the sample design waveform.



Results

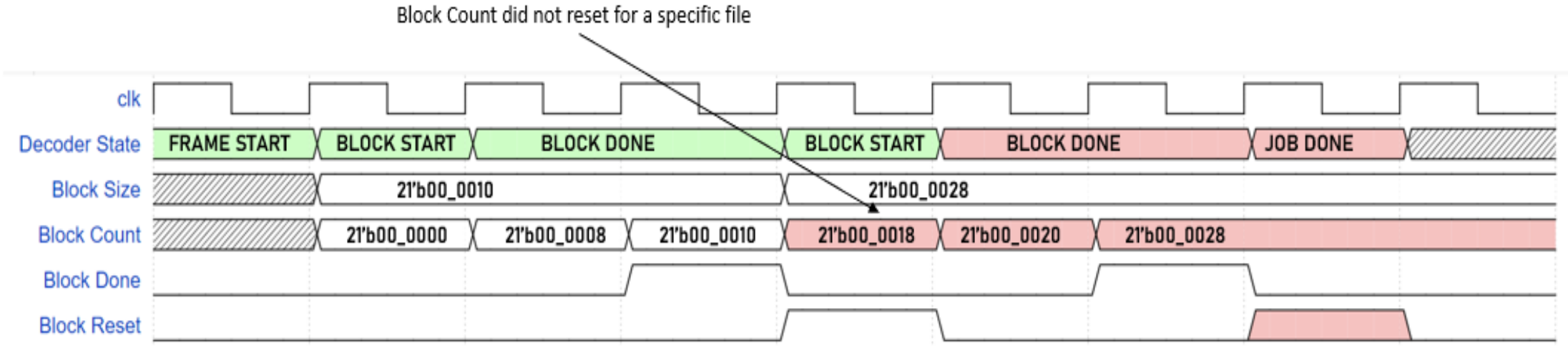
1. **Innovative Strategy using Instant Inspection**
   The simulation team leveraged bug scenarios discovered by FPV to develop targeted tests
2. **Enabled recreation of bugs in a controlled simulation environment**
3. **Validation of bugs and hypothesis for checkpoints**
   Validated the existence of these bugs, facilitated a detailed understanding of their behavior and validated hypothesis for checkpoints



Corner case bug #1

Block Reset erroneously asserted:

- **Hypothesis Formulation:** "BLOCK_RESET" signal must be asserted whenever the decoder has completed processing the current block and starts processing the next block.
- **Bug Detection:** This hypothesis underwent multiple iterations, resulting in several failures, before finally detecting a buggy case for a specific compressed file.



Conclusion

INNOVATIVE STRATEGY

Instant inspection with hypothesis method and co-use of simulation and FPV highlights an innovative way to validate stream decoders

EFFECTIVENESS OF FORMAL

Results demonstrate the effectiveness of FV approach for stream decoders in modern SoCs

INCREASED CONFIDENCE

The findings increase confidence in FV as a robust tool for ensuring the accuracy and dependability of stream decoders

BENEFITS OF ADOPTION OF FORMAL

- Improved system performance
- Reduced risk of data corruption
- Enhanced user experiences across various applications

Q&A

accellera
SYSTEMS INITIATIVE