2023
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
FEBRUARY 27-MARCH 2, 2023

Deadlock Free Design Assurance Using Architectural Formal Verification

Bhushan Parikh, Shaman Narayana

Intel Corporation

accellera
SYSTEMS INITIATIVE

Agenda

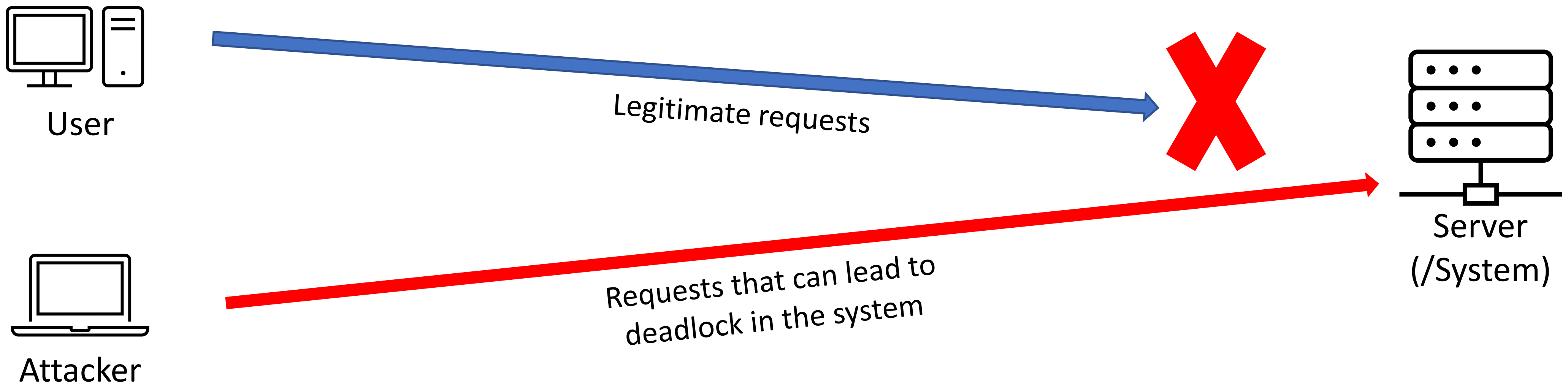
- Problem statement
  - Prove no deadlock in the system
- Example of a complex system
  - Lossless HW Compression IP
- Challenges in verifying no deadlock requirement
- Architectural Formal Verification overview
- Results and case studies
- Summary

Acronyms

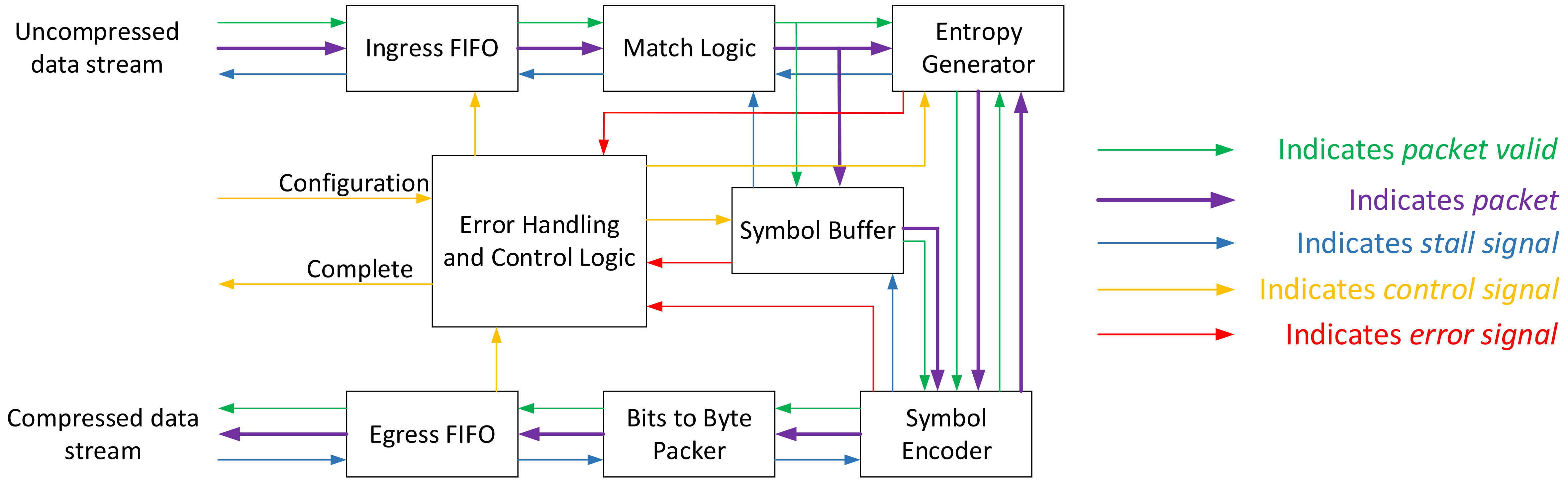
• Si	=>	Silicon
• FV	=>	Formal Verification
• PSV	=>	Pre-Si Verification
• AM	=>	Architectural Model
• SV	=>	System Verilog
• FIFO	=>	First In First Out
• AFV	=>	Architectural Formal Verification

Problem Statement

- Prove no deadlock in the design
  - A deadlock in the system can lead to a Denial of Service (DoS) attack
  - PSV requires running many test cases targeting very specific conditions
  - Standard(/Traditional) FV methods are not effective at system level due to exponential complexity



Lossless HW Compression IP: An Example



No deadlock in the design == Lossless HW Compression IP must assert completion signal for every end of the request packet irrespective of any error

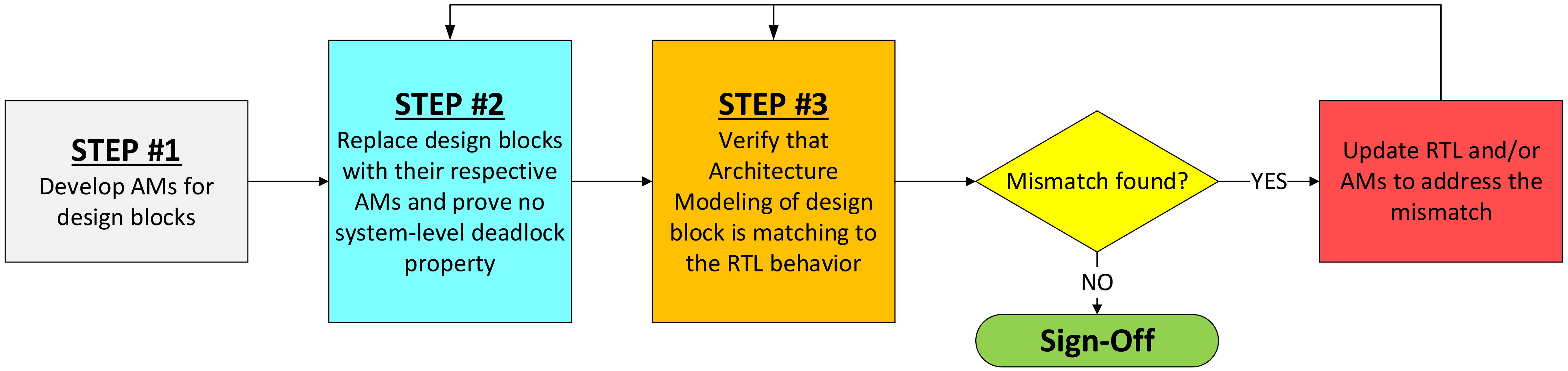
Challenges Associated with PSV

- Full sign-off is not feasible
  - Exponentially large number of input combinations
  - Verification of all input combinations are required
  - Error scenarios further increases the test cases

Challenges Associated with Traditional FV

SVA Property #	SVA Property	Status	Bound #	Time
Assert 1	A completion signal is asserted for every end of the request packet	Undetermined	15	~ 24 hours
Cover 1	A completion signal is asserted for end of the request packet	Covered	25	< 5 minutes
Cover 2	Input packet is sent from Ingress FIFO to Match Logic	Covered	6	< 1 minute
Cover 3	Match Logic processed input and sent output to Entropy Generator	Undetermined	50	~ 24 hours
Cover 4	Entropy generation is complete	Undetermined	38	~ 24 hours

Proposed Methodology: Architectural FV



STEP #1
Develop AMs for design blocks
STEP #2
Replace design blocks with their respective AMs and prove no system-level deadlock property
STEP #3
Verify that Architecture Modeling of design block is matching to the RTL behavior
Mismatch found?
YES
Update RTL and/or AMs to address the mismatch
NO
Sign-Off

Step1: Develop AMs

- Model the system-level requirements using the SV *ASSUME* properties
- Abstract requirements not applicable to the system-level requirement

Ingress FIFO AM Example: Requirements

1. In the absence of an error, the FIFO must forward packets to the Match Logic
2. The FIFO must drop all incoming packets except the end of the request packet in case of an error
3. The FIFO must drop all subsequent packets after the completion signal is asserted
4. FIFO must not overrun or underrun
5. Value of the data length must match with valid data bytes in the packet
6. The FIFO must not corrupt the data

Control specific

Data specific

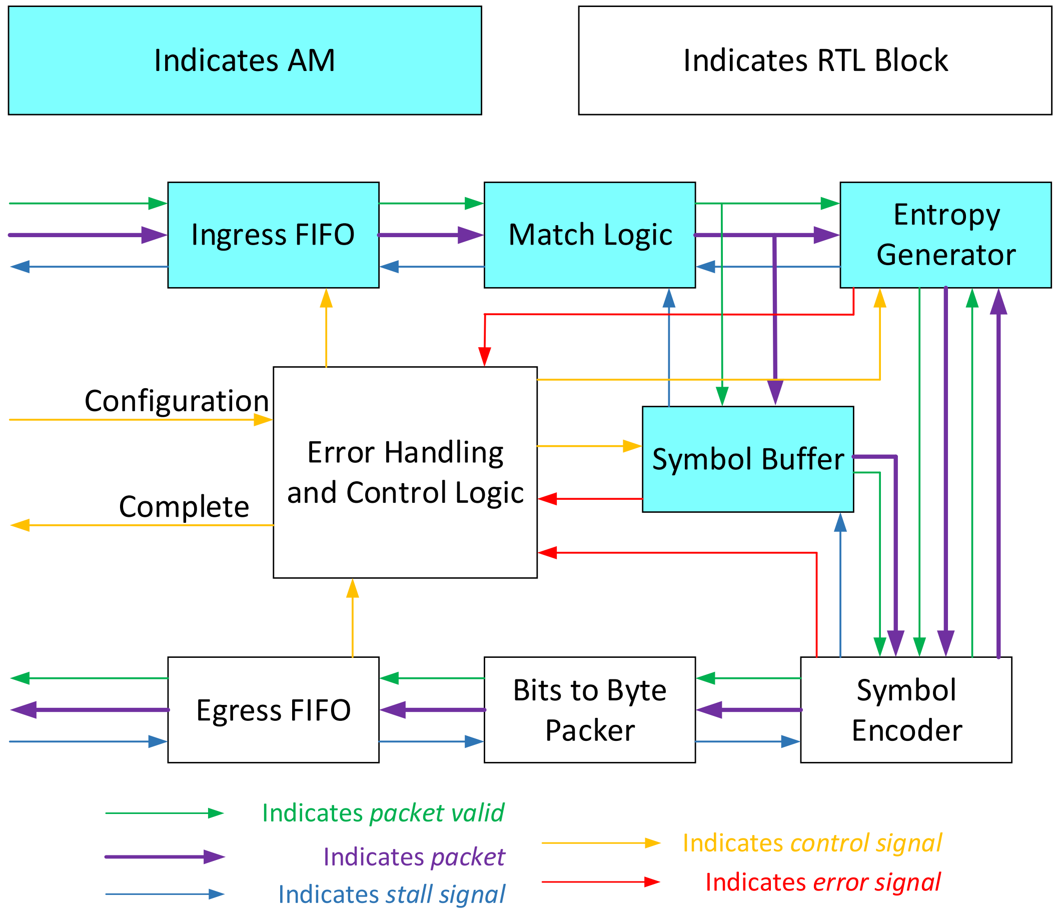
Ingress FIFO AM Example: SV Properties

- Modeling of the data is not required
- Modeling of the data length is not required

```
module Ingress_FIFO (
                         //input
                         input [000:000] packet_in_valid,
                         input [128:000] packet_in_payload,
                         input [003:000] packet_in_len_bytes,
                         input [000:000] error,
                         input [000:000] completion,
                         input [000:000] output_bigger_than_expected,

                         //output
                         output [000:000] packet_out_valid,
                         output [063:000] packet_out_payload,
                         output [003:000] packet_out_len_bytes,
                         output [000:000] get_next_packet,
                         output [000:000] full
                    );

    localparam END_OF_REQUEST_PACKET = 15;

    assume –name forward_packets_if_no_error
                    {in_fifo_vld_pkt_in & ~error |=> ##[0:5] in_fifo_vld_pkt_out}

    assume –name drop_packets_if_errors_or_completion_or_biger_output
                    {$rose(error | completion | output_bigger_than_expected) |->
                         (~packet_out_valid & get_next_packet) s_until (packet_payload[3:0] == END_OF_REQUEST_PACKET)}

    assume –name no_input_packets_if_ingress_fifo_is_full
                    {full |-> ~get_next_packet}

    assume –name send_end_of_request_packet_if_error_or_bigger_output
                    {$rose(error | output_bigger_than_expected) |=> (packet_out_payload[3:0] == END_OF_REQUEST_PACKET)}

endmodule
```

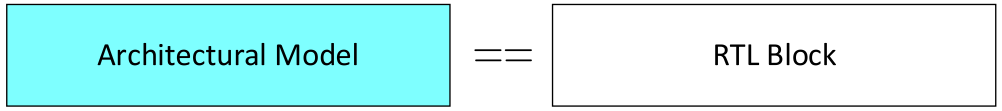
Step2: Replace RTL Blocks with AMs

- Leave the RTL block(/s) with simple functionality as it is
  - Symbol Encoder
  - Bit to Byte Packer
  - Egress FIFO
- No modeling for key control RTL block(/s)
  - Error Handling and Control Logic
- Prove the system-level requirement of no deadlock



Step3: Verify AMs and RTL Blocks

- Verify that the SV properties used for modeling the AMs are matching the RTL behavior
  - The ASSUME properties of the AMs become ASSERT properties for each block
  - Leverage traditional FV methodology
  - Less complexity due to block-level FV
  - Utilize design reduction techniques if required



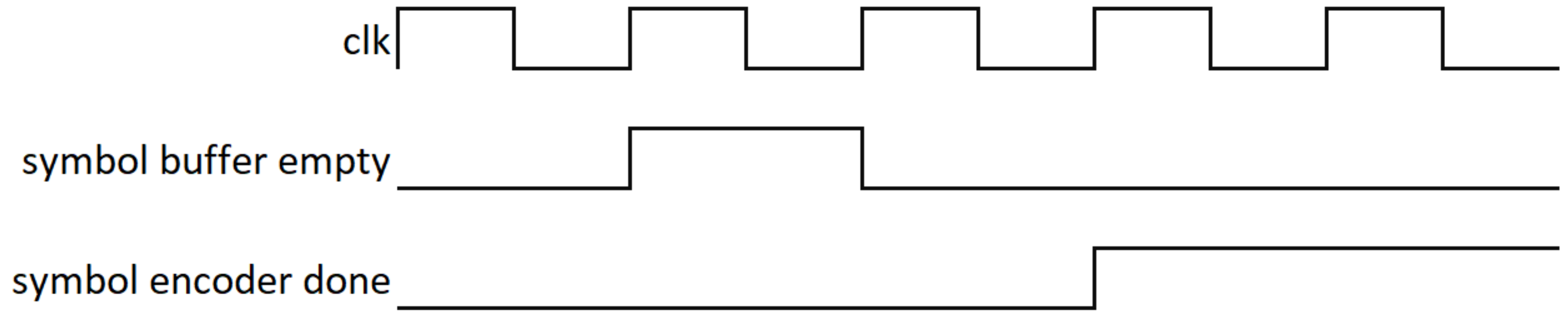
Results

SVA Property #	SVA Property	Status	Bound #	Time
Assert 1	A completion signal is asserted for every end of the request packet	Converged	52	< 30 minutes
Cover 3	Match Logic processed input and sent output to Entropy Generator	Covered	75	< 5 minutes
Cover 4	Entropy generation is complete	Covered	84	< 10 minutes

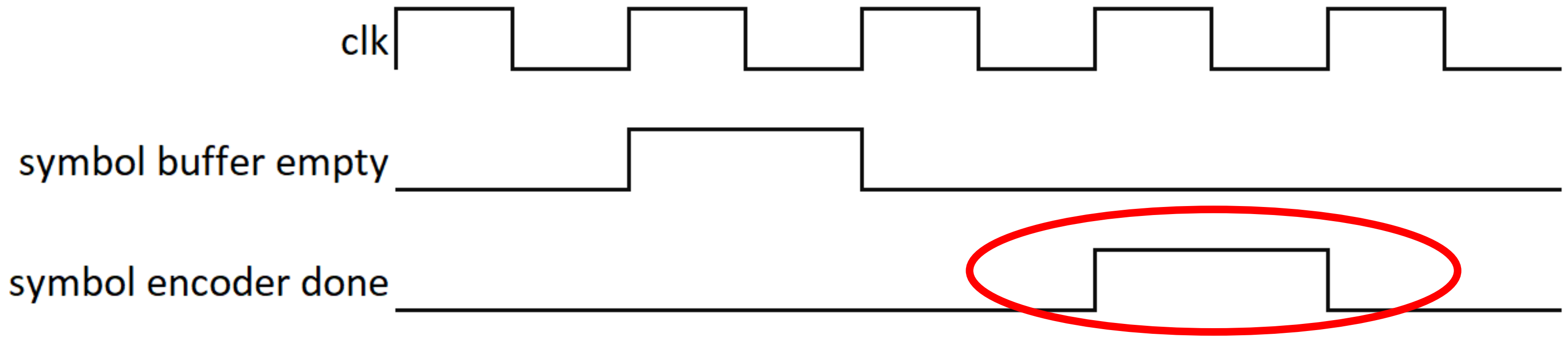
- Identified multiple logic defects in the PSV signed-off design
  - Required complex stimulus and alignment of multiple events in the PSV environment
  - Identified improvements in the design for error-related cases

Logic Defects: Case Study 1

- Expected behavior
  - Observed in all PSV test cases

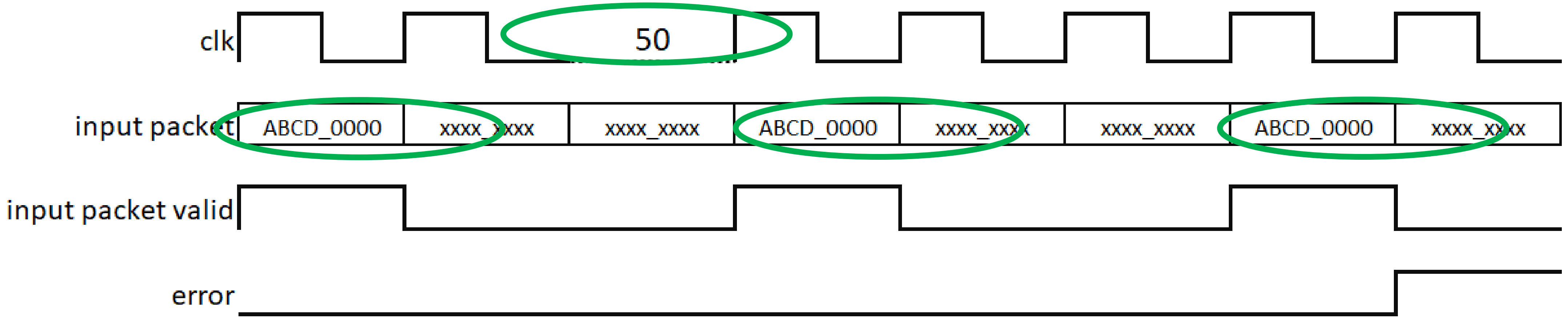


- Failing Scenario
  - Required specific combination of the input starvation and output back-pressure

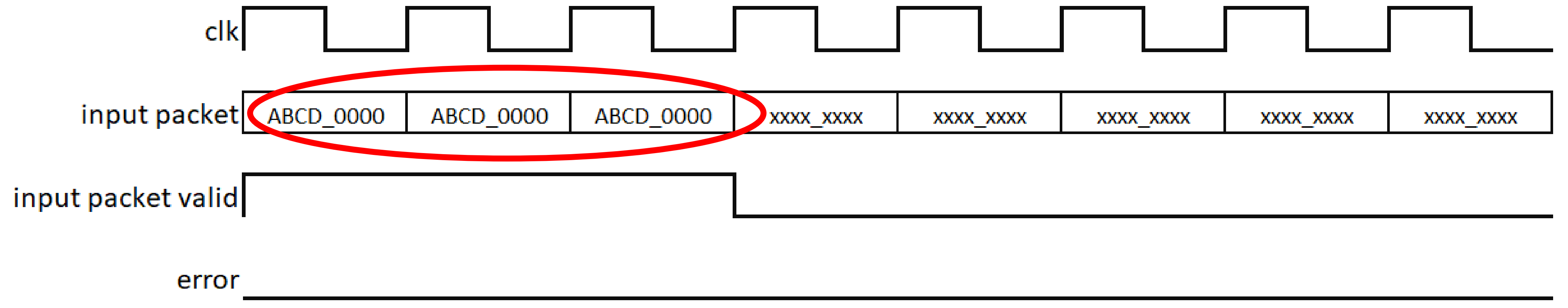


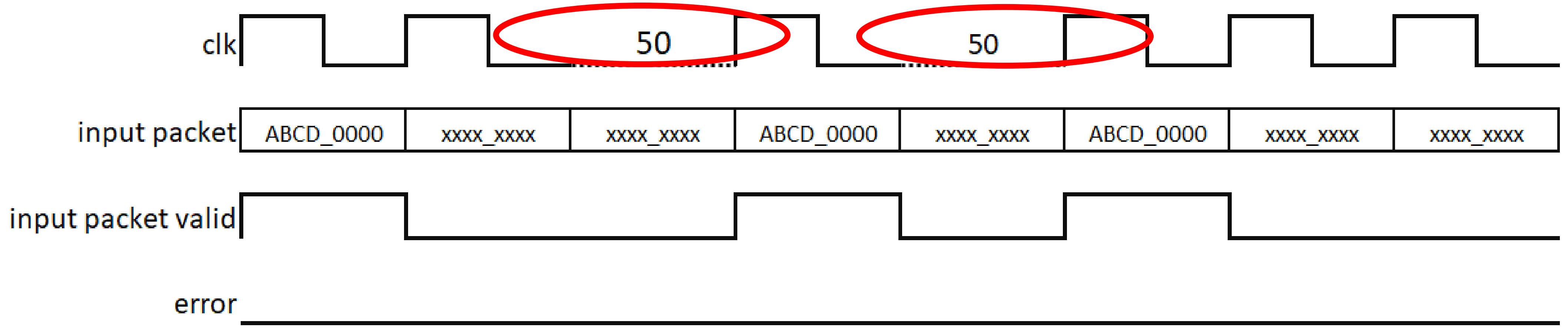
Logic Defects: Case Study 2

- Error and Control FSM did not comprehend the handling of all errors
- Missing error scenario required a specific combination of input pattern and delay



Case Study 2: Behavior Observed in PSV Runs





Case Studies: Further Analysis

- Case study 1:
  - Updated PSV test-bench
  - Reliably reproduced the logic defect in PSV within 1 month of execution

- Case study 2:
  - Why missed in the PSV coverage review?
    - Error injection was used in PSV
    - Injection led to multiple error assertions
  - Updated PSV test-bench
  - Reliably reproduced the logic defect in PSV after 4 months of execution

Learnings

- Advance planning is required for AFV
- Proving the system-level requirement is not enough
- May help identify gaps in the architecture

Summary

- Proving No deadlock at the system-level is practical using AFV
- Deployed the AFV methodology for a complex system and discovered all the logic defects
- AFV is a highly scalable methodology

Thank You ☺

Questions?