

# The Design is not Correct!

Adam Krolnik

Co-Author *Assertion-Based Design* 2<sup>nd</sup> Edition

ZSP Verification Mgr. LSI Logic Corp

# Introduction

- Assertions are 50+ years old [Alan Turing]
- Conceived to assist in validation
- Statements of truth to be verified.
- Basis for functional coverage.

# Designer and DV Tool

- Allows Concise description of protocols
- Easily written next to HDL code for review, recall, etc.
- Describe intent not normally known to DV!
- Trap faulty behavior closer to source.
- Quick to modify if spec. or behavior is changed.
- Minimal effort (few hours) for good results

# Best practices

- **Errors get fixed**, **Warnings are ignored**.
- *Thinking* about assertions can find bugs.
- Write assertions for module interfaces *as they are written*. [**~1 per port**] (HW)
- Write protocol checks/monitors (DV)
- Minimum **#3-5** required as encouragement. (HW)
- Active comments detect bad behavior (HW)

# Best Practices cont.

- Verify old vs. new equations (HW timing opt)

- Error messages convey **missing** information  
Illegal alignment of address (25) for **word size**

Illegal request during pending flush, See **John Extra done sent. See BR block.**

**Request (4) did not complete in 1000 cycles.**

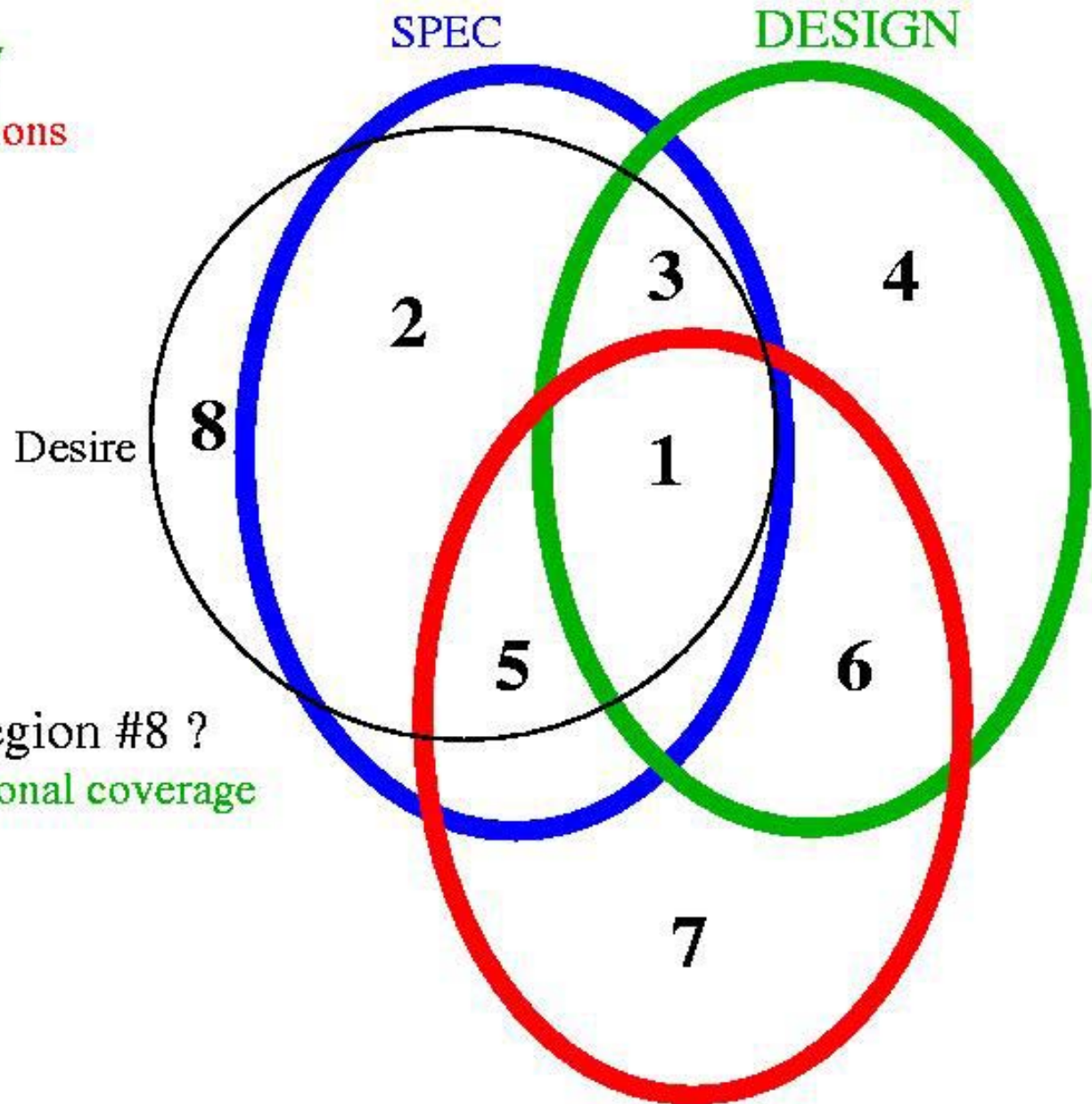
- **Stop** simulations on the first error.

- **Add** assertions as you find problems

Black box DV

Traditional DV

Designer Assertions



What about region #8 ?

Target of functional coverage

Regions of Intent [Piziali and Wilcox]

HDL  
(Implementation)

# OVL vs. PSL vs. SVA

- 80% of Assertions are simple
  - SVA** `assert property (@(posedge clk) disable_iff !rst_n ( expression ));`
  - PSL** `assert always (expression) @(posedge clk) abort !rst_n ;`
  - OVL** `assert_always A1(clk, rst_n, expression );`
- 10% of Assertions need either
  - SVA** `not ( expr1 ##1 expr2) or expr1 |-> expr2`
  - PSL** `( expr1 ; expr2) or expr1 |-> expr2`
  - OVL** `assert next()`

# Next steps

- Protocol verification tools
  - Need assertions on interface to define legal behavior (**best practice**)
- End to end verification
  - Need well defined specifications  
(**Assertions help reduce ambiguity**)

# Conclusions

- **Unbug** the design, don't **debug** the design.  
**Assertions** are the designers part.
- Assertions are concise;  
simplifying monitors and checking code
- Focus assertions on interfaces to trap wrong behavior early.