

# **Is it time to declare a verification war?**

**Brian Bailey**

Email: [brian\\_bailey@acm.org](mailto:brian_bailey@acm.org)

Tel: 503 632 7448

Cell: 503 753 6040

Web: [brianbailey.us](http://brianbailey.us)

# In the beginning



- The year app. 500 BC
- Sun Tzu published "on the art of war"  
孫子兵法
- Since then it has been used for
  - **Military strategies**
  - **Political**
  - **Business**
  - **Anything requiring tactics**

# Why was it so important

- **Sun Tzu was the first to recognize the importance of positioning in strategy and that position is affected both by:**
  - **objective conditions in the physical environment**
  - **subjective opinions of competitive actors in that environment**



- **He taught that strategy was not a to do list**
  - **requires quick and appropriate responses to changing conditions**
- **Begins to sound like verification**
  - **Can we learn from Sun Tzu?**

# Who is the Enemy?



- **When thinking about verification, the battle is being fought against Murphy the Design**
  - **Murphy is powerful and indiscriminate**
  - **He will show up anywhere and attempt to cause the maximum damage**
  - **He is also indestructible**
  - **Kill him and he shows up somewhere else**
  - **Is it possible to know he is dead – no we know he will never die**
- **Why then do we bother fighting?**
  - **Because we are the verification heroes!**

# Sun Tzu's 13 Chapters

- 1. Laying Plans** explores the five key elements that define competitive position (mission, climate, ground, leadership, and methods) and how to evaluate your competitive strengths against your competition.
- 2. Waging War** explains how to understand the economic nature of competition and how success requires making the winning play, which in turn, requires limiting the cost of competition and conflict.
- 3. Attack by Stratagem** defines the source of strength as unity, not size, and the five ingredients that you need to succeed in any competitive situation.
- 4. Tactical Dispositions** explains the importance of defending existing positions until you can advance them and how you must recognize opportunities, not try to create them.
- 5. Energy** explains the use of creativity and timing in building your competitive momentum.
- 6. Weak Points & Strong** explains how your opportunities come from the openings in the environment caused by the relative weakness of your competitors in a given area.
- 7. Maneuvering** explains the dangers of direct conflict and how to win those confrontations when they are forced upon you.
- 8. Variation in Tactics** focuses on the need for flexibility in your responses. It explains how to respond to shifting circumstances successfully.
- 9. The Army on the March** describes the different situations in which you find yourselves as you move into new competitive arenas and how to respond to them. Much of it focuses on evaluating the intentions of others.
- 10. Terrain** looks at the three general areas of resistance (distance, dangers, and barriers) and the six types of ground positions that arise from them. Each of these six field positions offer certain advantages and disadvantages.
- 11. The Nine Situations** describe nine common situations (or stages) in a competitive campaign, from scattering to deadly, and the specific focus you need to successfully navigate each of them.
- 12. The Attack by Fire** explains the use of weapons generally and the use of the environment as a weapon specifically. It examines the five targets for attack, the five types of environmental attack, and the appropriate responses to such attack.
- 13. The Use of Spies** focuses on the importance of developing good information sources, specifically the five types of sources and how to manage them.

# The Key Essence

- **Know the enemy**
  - **Understand the design and the best ways to approach the verification challenge**
- **Know yourself**
  - **Understand (and improve) the process**
  - **Understand its strengths and weaknesses**
- **Prepare yourself**
  - **Make sure the tools you use are those most likely to lead to success**
  - **Be flexible in the way that you use them**
- **Use feedback**
  - **Based on objective observations**

# Outline

- **Know yourself**
  - **I am going to revisit some fundamentals of verification that many forget**
  - **Take a look at coverage metrics**
  - **Primarily for the people newer to verification**
    - **Also a good reminder to more seasoned amongst us**
- **Arm yourself with the best weapons**
  - **Some recent tools that help**
    - **Make verification more objective**
    - **Raise the level of abstraction**
    - **Preserve and use knowledge**

# Verification definition

***"Confirmation by examination and provisions of objective evidence that specified requirements have been fulfilled."***

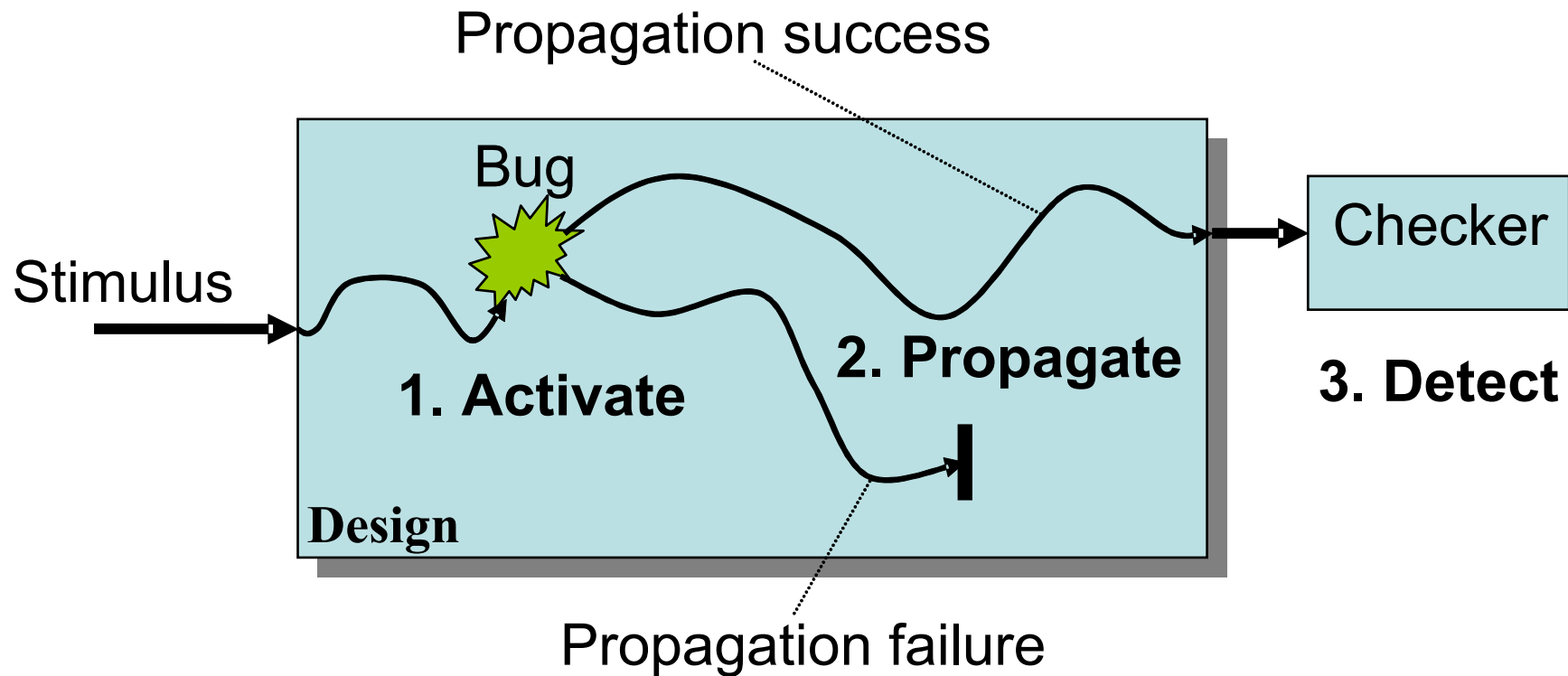
*IEEE Definition of verification.*

- ▶ **Verification is all about answering the question:**  
**Have we implemented something correctly ?**

# Verification definition

- **4 Key phrases**
  - ***Confirmation by examination***
    - If you don't look, then there is no hope of finding incorrect behavior
  - ***provisions of objective evidence***
    - Requires a second independent model of functionality
  - ***specified requirements***
    - This introduces the needs for coverage to ensure the right things have been verified
  - ***have been fulfilled***
    - Requires an act of verification to be associated with a coverage measurement

# Verification fundamentals



**Remember:** If you don't look it hasn't been verified  
If it hasn't been verified against something objective, then it isn't trustworthy

**Only place this happens is in the checker**

# Fundamentals - Coverage

- **The extent or degree to which something is observed, analyzed, and reported.**

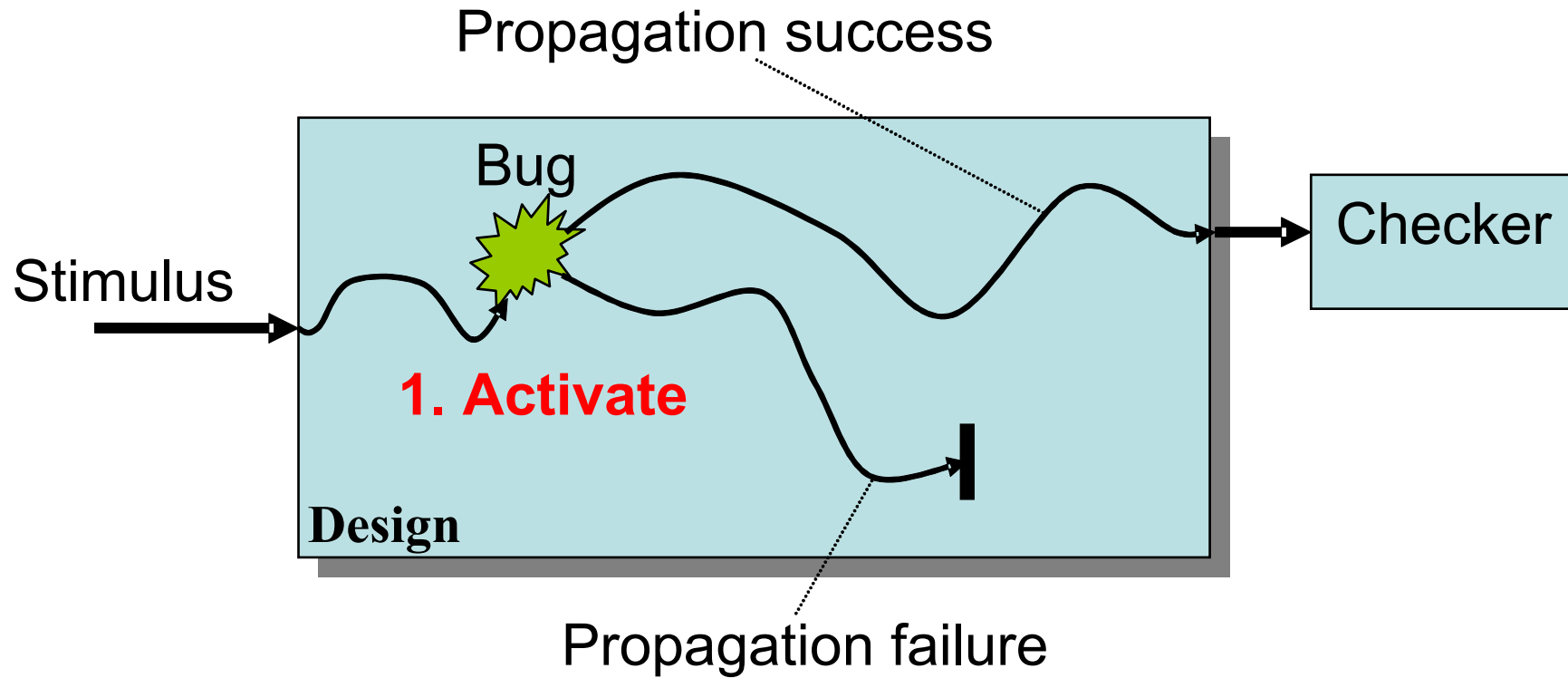
thefreedictionary.com

- **High coverage does not imply high quality**
  - **With the metrics that are in use today**
- **While coverage metrics are objective (the information they provide is impassionate)**
  - **The decision about which to apply is subjective**
  - **The analysis of results is often subjective**
  - **Most coverage metrics do not provide “*Confirmation by examination and provisions of objective evidence*”**
    - ***This include code coverage, functional coverage and almost all other metrics in use***

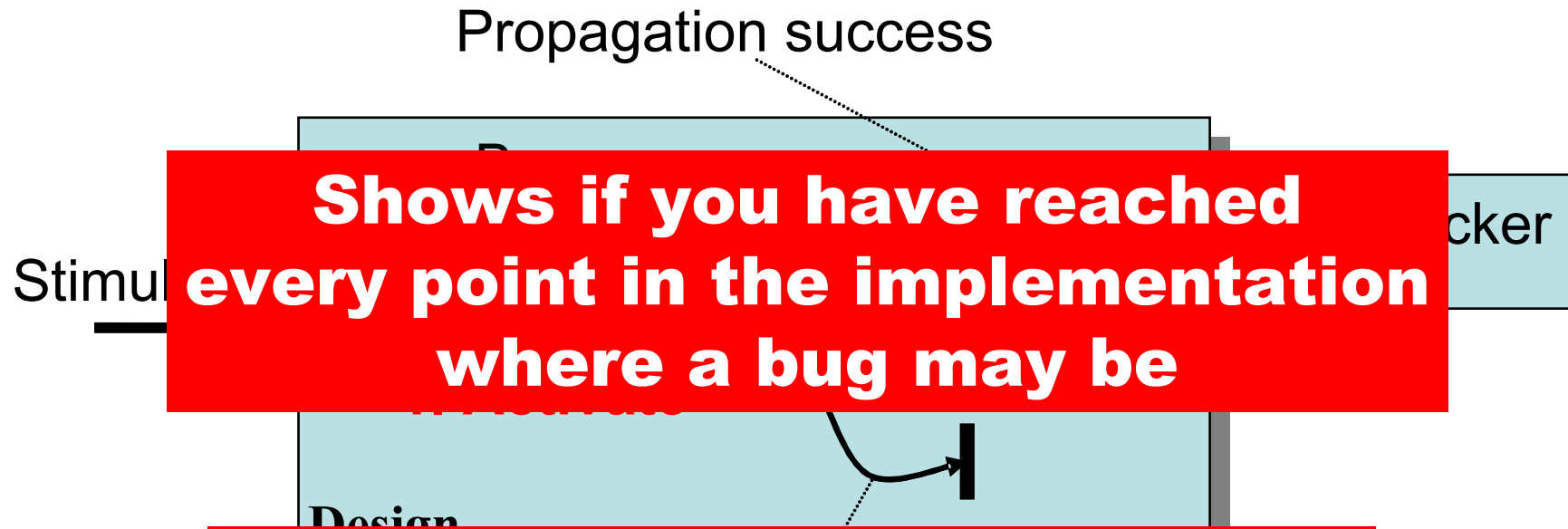
# Structural Metrics

- **These metrics are automatically extracted from the implementation source**
  - ✓ **Easy to implement**
  - ✗ **Cannot identify what is missing**
- **They tell you that something was 'reached'**
  - **A line of code**
  - **A decision point**
  - **A state**
- ✗ **They do not tell you that it was reached for the right reason**
- ✗ **They do not tell you that the right thing happened after that**

# Structural Coverage



# Structural Coverage



**Shows if you have reached every point in the implementation where a bug may be**

**Does not show you that the bug would have been detected**

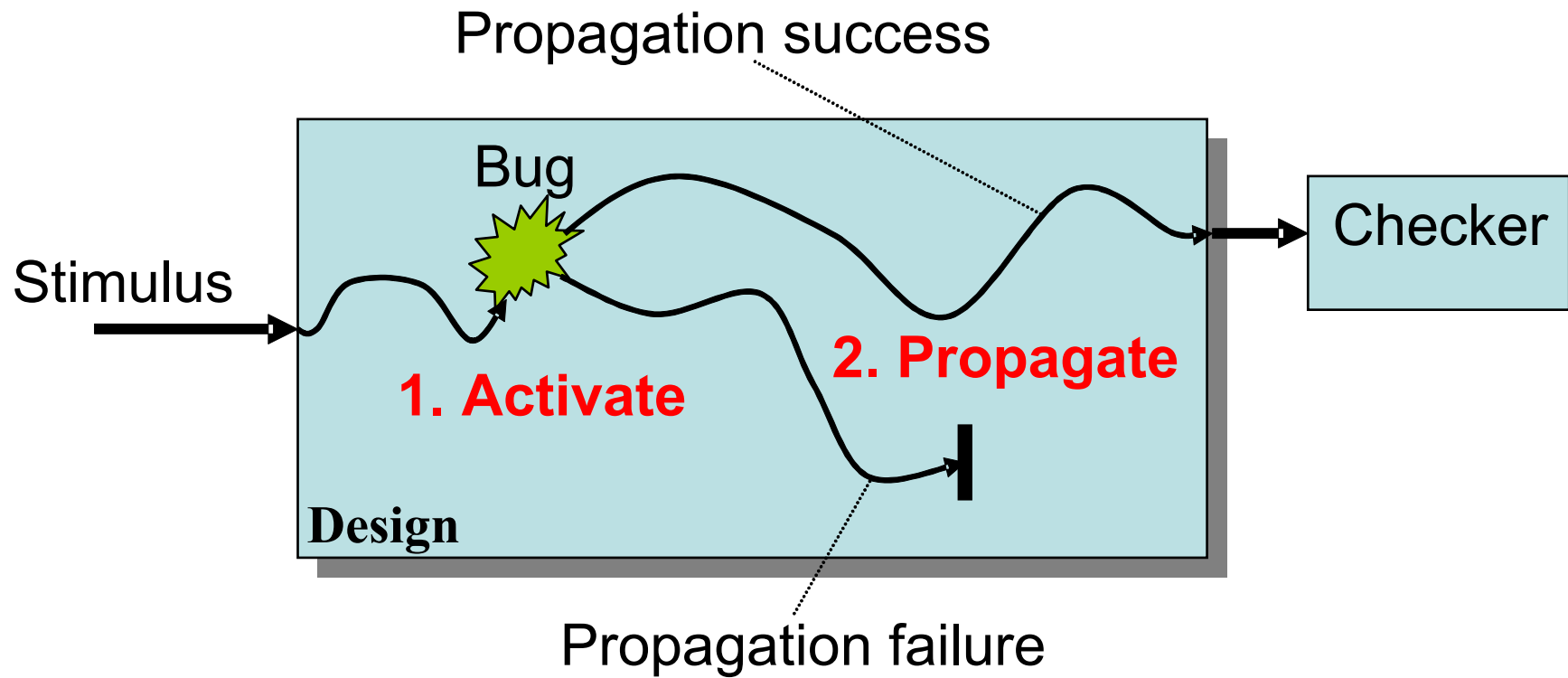
**Coverage != Verification**

# Structural metrics

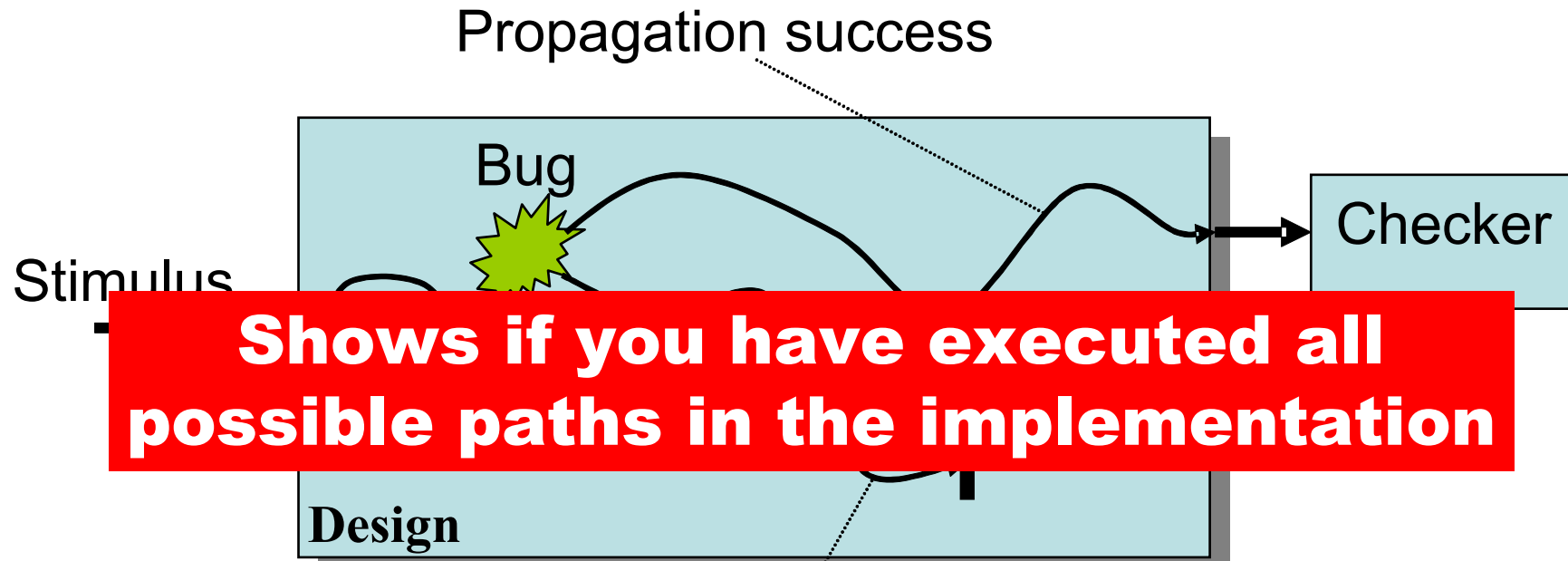
- **Path coverage**

- ✓ **This is the set of all combinations of all branches**
- ✓ **It identifies exhaustive execution of a model**
- ✗ **Still cannot identify missing functionality**
- ✗ **Expensive computationally**
  - **Limited path sets have been defined but not in use**
- ✗ **Does not identify data errors that do not affect control**
  - **Would not have identified Intel FP bug since this was related to values in a ROM**

# Path Coverage



# Path Coverage



**Shows if you have executed all possible paths in the implementation**

**Does not show you that the bug would have been detected**

# Structural coverage – dirty word

- **When did structural coverage become a dirty word?**
  - Directed tests target specific functionality
  - structural coverage was an impartial way to identify holes
- **Along came constrained/pseudo-random generation**
  - No longer targeting specific functionality
  - Needed a way to ensure important functionality was executed
- **Thus functional coverage was born**
  - There is nothing wrong with structural coverage coupled to directed testing
    - Unless used irresponsibly
    - Becoming less useful with increased concurrency

# Functional Coverage

- **Identifies indicators of functionality**
  - **A data value**
  - **A specific state**
  - **A sequence of states**
  - **etc**
- ☒ **It does not identify that the functionality is correct**
  - **Still expects that the comparison of two models is happening**
  - **Suffers from some of the same problems as code coverage**

# Functional coverage

- **It is a third independent model**
  - **Does not define the correct behaviors, just the behaviors that should be detectable**
  - **Different language**
    - **Good and bad**
  - No way to define completion**
    - **Implementation is subjective**
    - **Closest theoretical model is path coverage**
      - **Expensive in terms of execution time**
- **Higher cost than structural coverage**
  - An extra model to define**
  - Slows down simulator**
  - Analysis of holes is easier**

**Functional Coverage != Verification**

# Adding Propagation

- **Path Coverage**
  - **Already talked about this**
- **OCCOM** (Observability-based Code Coverage Metrics )
  - Srinivas Devadas Abhijit Ghosh Kurt Keutzer – DAC 1998
  - **Computes the probability that an effect of the fault would be propagated**
  - **During normal simulation – variables are tagged**
    - **Positive and negative tags are used**
  - **In a post processing step, these tags are used to compute the probabilities of propagation**

# OCCOM results

Ex	#Tags	Directed				Random			
		#Vec	#Observed Tags	Tag Coverage%	Line Coverage%	#Vec	#Observed Tags	Tag Coverage%	Line Coverage%
mult	44	109	38	86%	100%	25	34	77%	100%
pport	33	18	29	87%	92%	100	22	67%	90%
arbiter	60	88	56	93%	100%	5000	34	57%	90%
count	100	3000	68	68%	92%	3000	68	68%	92%
schsm	52	70	44	85%	94%	4300	28	54%	90%

- **Notes:**

- Shows the problems with coupling code coverage and random techniques
- Shows that people creating directed tests consider propagation
  - This is a flaw with random methods
- Some of the new intelligent testbench tools will make this a lot worse

# Summary of coverage methods

	Cost to implement	Cost to execute	Imp or spec	Completion	Objective	Benefits	Analysis
<b>Code</b>	<b>Low</b>	<b>Low</b>	<b>Imp</b>	<b>Low</b>	<b>Yes</b>	<b>Low</b>	<b>Difficult</b>
<b>Path</b>	<b>Low</b>	<b>Low</b>	<b>Imp</b>	<b>High</b>	<b>Yes</b>	<b>High</b>	<b>Moderate</b>
<b>Toggle</b>	<b>Low</b>	<b>Low</b>	<b>Imp</b>	<b>Medium</b>	<b>Yes</b>	<b>Medium</b>	<b>Moderate</b>
<b>Functional</b>	<b>Medium</b>	<b>Medium</b>	<b>Spec</b>	<b>?*</b>	<b>No</b>	<b>Medium</b>	<b>Easy</b>
<b>Assertion</b>	<b>High</b>	<b>High</b>	<b>Spec</b>	<b>?*</b>	<b>No</b>	<b>High</b>	<b>Difficult</b>

**\* No direct way to measure today**

# Arm yourself with the best weapons

- A look at some new technologies
  - **Censored** product from Jasper
  - Raising abstraction with Calypto
  - Functional Qualification from Certess

# Using the right weapon

- **We have a number of different products in the verification portfolio**
  - **Each have very different characteristics**
  - **Each have different areas where they can excel**
  - **They also have weaknesses**
- **Need to carefully match the tool to the situation**
  - **Verification planning helps in this regard**
  - **Requires thought upfront**

# Thinking slightly differently

- **But it is not just the application of the tools**
  - **It is also the application of technology to make tools**
  - **For years formal methods were used to make formal verification tools**
    - **Seems obvious, but limited usefulness**
    - **Capacity constraints etc**
  - **Started to target technology to fit the problem**
    - **Bounded model checking**
    - **Semi-formal verification**
    - **Application of abstractions**
  - **Look for completely different applications**
    - **This is what I really wanted to talk about, but could not get permission from the company**

# Another example

- **Intelligent testbenches**
  - **Two primary types**
    - **Graph based**
    - **Simulation based**
  - **Graph based ones basically create a formal model of the system and use it to create stimulus paths that in turn exercise the design**
  - **Simulation based ones “learn” as they simulate so that they can get to an “objective” faster**
  - **Neither type of company wants to be called “formal” – but they both employ formal technologies**

# Raising the abstraction

- **Just as combinatorial equivalence checking allowed us to move from gate to RTL verification**
  - **Sequential equivalence checking will allow a migration to higher levels of abstraction**
    - **Higher simulation speeds**
    - **Enable longer runs**
    - **Use real scenarios (live data)**

# Uses for SEC

- **Untimed input in C/C++/SystemC**
  - **Ensure hand coded RTL functionally matches**
- **Synthesis verification**
  - **Ensure constraints, setup and options produced a valid result**
- **Power optimization**
  - **Clock gating**
  - **Power optimizations**



# Commercialization

- **Large pool of researchers**
  - **Shares many technologies with property checking**
- **Internally generated tools**
  - **IBM**
- **Esterel Studio – 2007**
  - **Only within Esterel environment**
- **Calypto introduced SLEC at DAC 2005**
  - **Enables retimed RTL to be verified**
  - **Untimed to timed functional equivalency**
  - **Integrated with Mentor, Forte, Cadence synthesis products**

# Mutation Analysis

- **Similar to manufacturing test**
  - **Looks for a change in values seen on an output**
  - **Stuck-at faults: what fault model is the equivalent for designer errors?**
    - **Mutation fault model based on two hypotheses:**
      - **that programmers or engineers write code that is close to being correct**
      - **that a test that distinguishes the good version from all its mutants is also sensitive to more complex errors**
    - **Potentially huge number of faults**
- **Concept introduced in 1971**
  - **First tool implementation in 1980**

# Mutation analysis

- **Performs complete stimulate and propagate analysis**
  - **Addresses --**
    - **If you don't look it hasn't been verified**
  - **But not –**
    - **If it hasn't been verified against something objective, then it isn't trustworthy**
- **This is the same as manufacturing test**
  - **It is not good enough to know that something was different**
  - **Must be able to detect that it was in error**

# Functional Qualification

- **Based on mutation analysis**
- **Several differences:**
  - **Functional Qualification includes the detection phase. For a fault to be *detected* there must be a check made so that at least one testcase fails**
  - **Functional qualification does not depend on propagation to a primary output. Directly supports white box assertions**
  - **Uses very different fault injections schemes to provide relevant results faster**
  - **Applied to hardware instead of software**

# Some other differences

- **For SW, mutation analysis was used to “cover” the program**
- **Functional qualification is being used as a quality measure for the testbench**
  - **First time verification engineers have a tool that allows them to objectively measure themselves**
- **Statistical in nature**
  - **Technology advancements to improve performance**
  - **Do not have to run all faults or all testcases**
  - **Stop as soon as a testbench flaw has been revealed**
  - **Tackle the difficult problems first, and let the easy ones take care of themselves**

# Certitude Metrics - ST References

## **Global Metric**

- ◆ Representing the overall quality of the Verification Environment
- ◆ ST reference : 75%, but usually higher

## **Activation Score**

- ◆ Measures the ability of the test suite to exercise all the RTL of the IP
- ◆ Similar to code coverage
- ◆ ST reference : 95%, & 100% explained
- ◆ Missing % should deeply studied & fixed or explained

## **Propagation Score**

- ◆ Measures the ability of the test suite to propagate mutations to the outputs of the IP
- ◆ ST reference : 80%, but should probably be enhanced by adding more test scenarios to reach 90%

## **Detection Score**

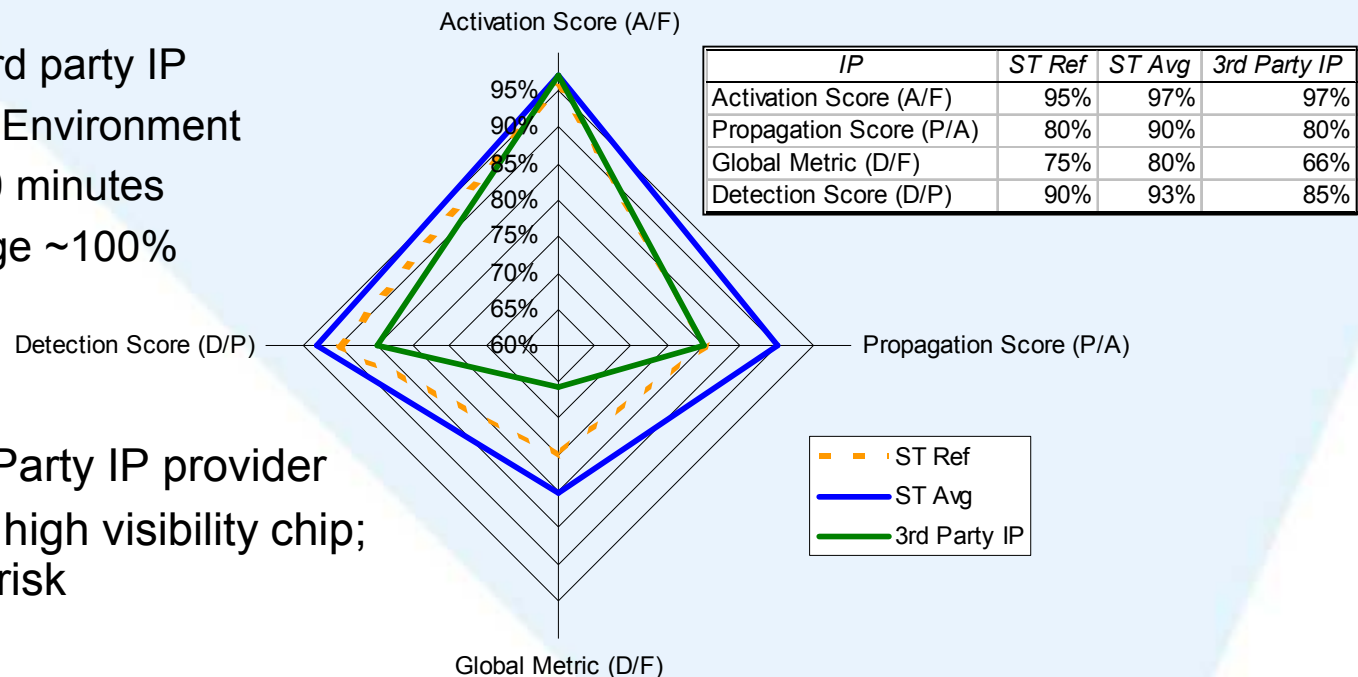
- ◆ Measures the ability of the environment to catch errors
- ◆ ST reference : 90%, but usually higher



# Case study 1 : 3rd Party - IP qualification

- **Case study 1:**

- Application: 3rd party IP
- HDL Directed Environment
- ~300 tests, 30 minutes
- Code Coverage ~100%



- **Challenges**

- Convince 3rd Party IP provider
- High revenue, high visibility chip; reduce respin risk

- **Results**

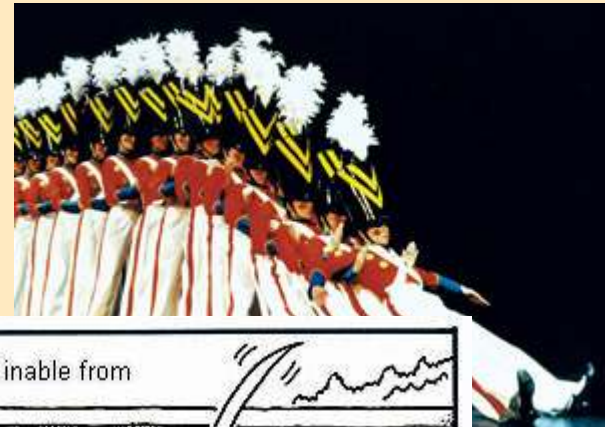
- Helped us to push IP provider to improve verification environment
  - and monitor progress
- Low detection score highlighted manual waveform checks



# Update from Haifa

- **Panel session: Coverage Metrics across the Verification Domain**
  - **Participants from SW, HW, industry, tool vendor, academia, dynamic and formal**
  - **3 out of 6 panelists identified mutation analysis as a key enabler**
  - **Holds promise as a way to make coverage objective**
  - **Holds promise as a way to integrate formal and dynamic methods**

# Is it “The Art of Verification”



Foolishly, the invisible man accepted the blindfold.

# Verification is Not Art

- We denigrate ourselves by calling it **Art**
- While we have not yet formalized and perfected the philosophy of verification
  - We do create highly adaptive strategies
  - We do use highly sophisticated tools
  - We do not believe that defeat is inevitable

The last word goes to Sun Tzu:

**To subdue the enemy without fighting is the supreme excellence**

The background features a repeating pattern of hexagons in shades of yellow, orange, and grey. A semi-transparent globe is positioned in the lower right quadrant, partially overlapping the hexagonal pattern.

# **Thank You**

# **Questions?**

**Email: [brian\\_bailey@acm.org](mailto:brian_bailey@acm.org)**

**Tel: 503 632 7448**

**Cell: 503 753 6040**

**Web: [brianbailey.us](http://brianbailey.us)**