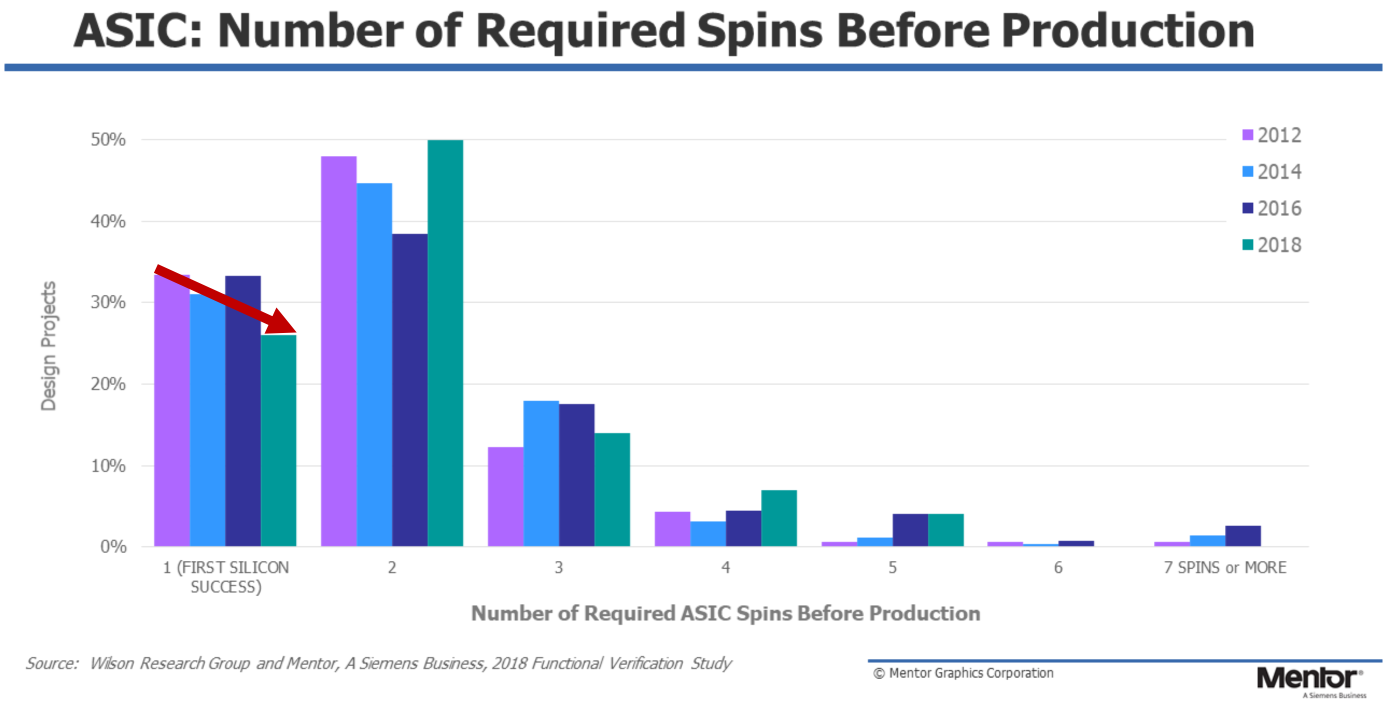
HSI Issues – Fake News or Not?

How Hardware/Software Interface Impacts Tape-Outs

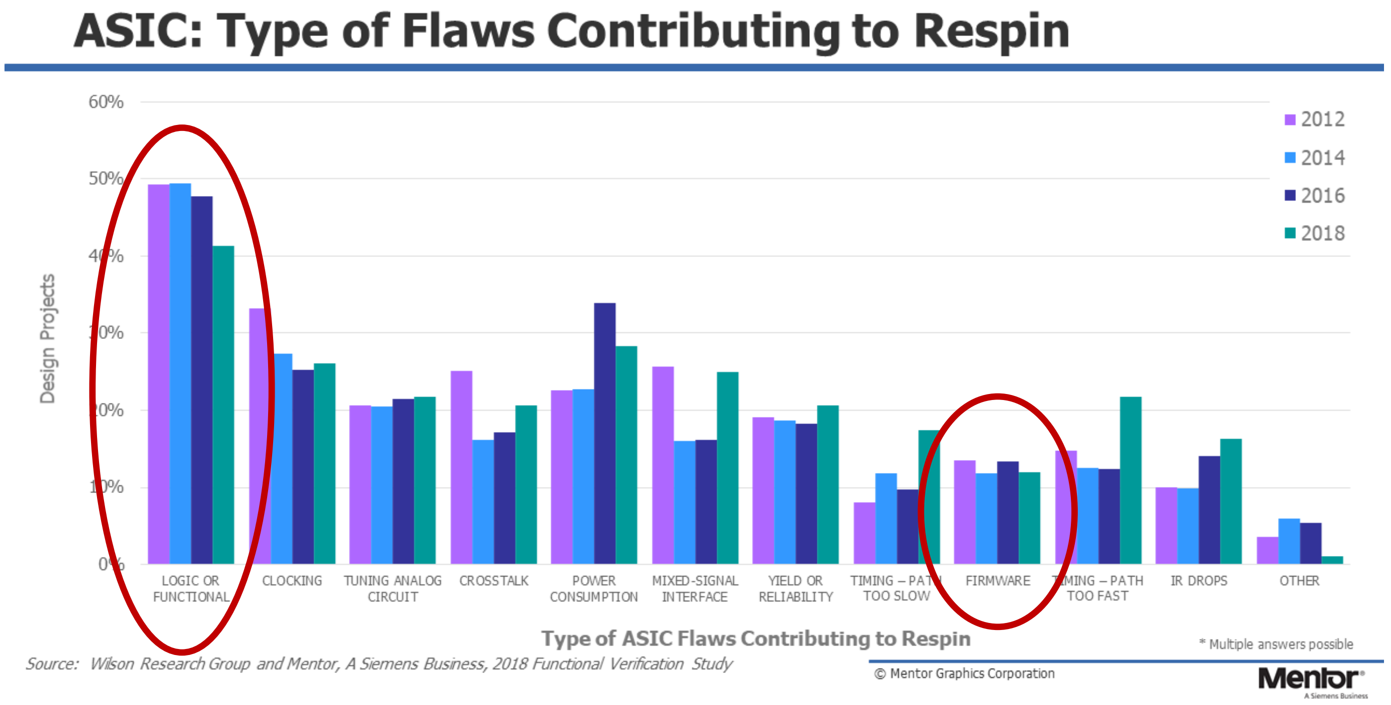
Gary Stringham - *Gary Stringham & Associates, LLC*
Rich Weber - *Semifore, Inc.*
Jamsheed Agahi - *Semifore, Inc.*

Wilson Study: Increasing Respins



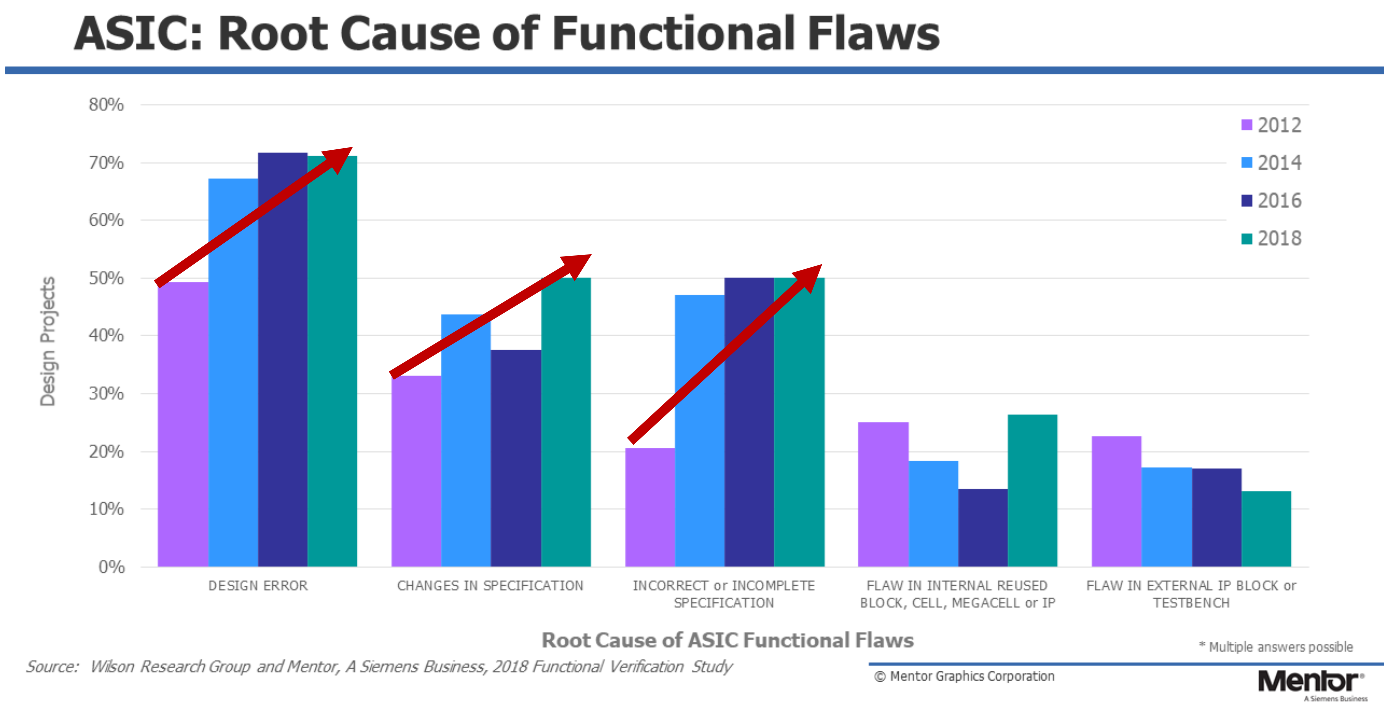
Source: Foster Harry, (2019) https://blogs.mentor.com/verificationhorizons/blog/2019/03/13/part-12-the-2018-wilson-research-group-functional-verification-study/

Logic/Functional, Firmware Flaws



Source: Foster Harry, (2019) https://blogs.mentor.com/verificationhorizons/blog/2019/03/13/part-12-the-2018-wilson-research-group-functional-verification-study/

Design Error, Specs Changed/Bad

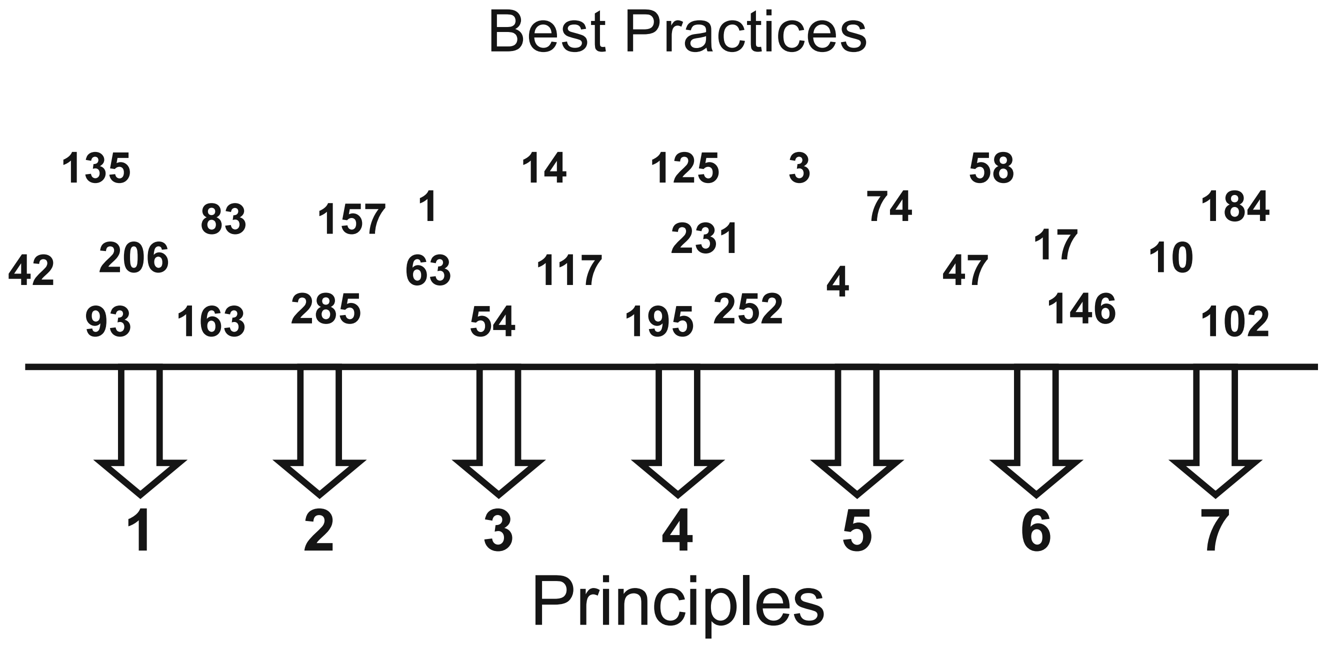


Source: Foster Harry, (2019) https://blogs.mentor.com/verificationhorizons/blog/2019/03/13/part-12-the-2018-wilson-research-group-functional-verification-study/

Improving HSI Will Reduce Respins

- Project Delays
- Chip Respins
- Cost Overruns
- Lost Productivity
- Inferior Quality

The Seven Principles of Successful Hardware/Software Interface Design



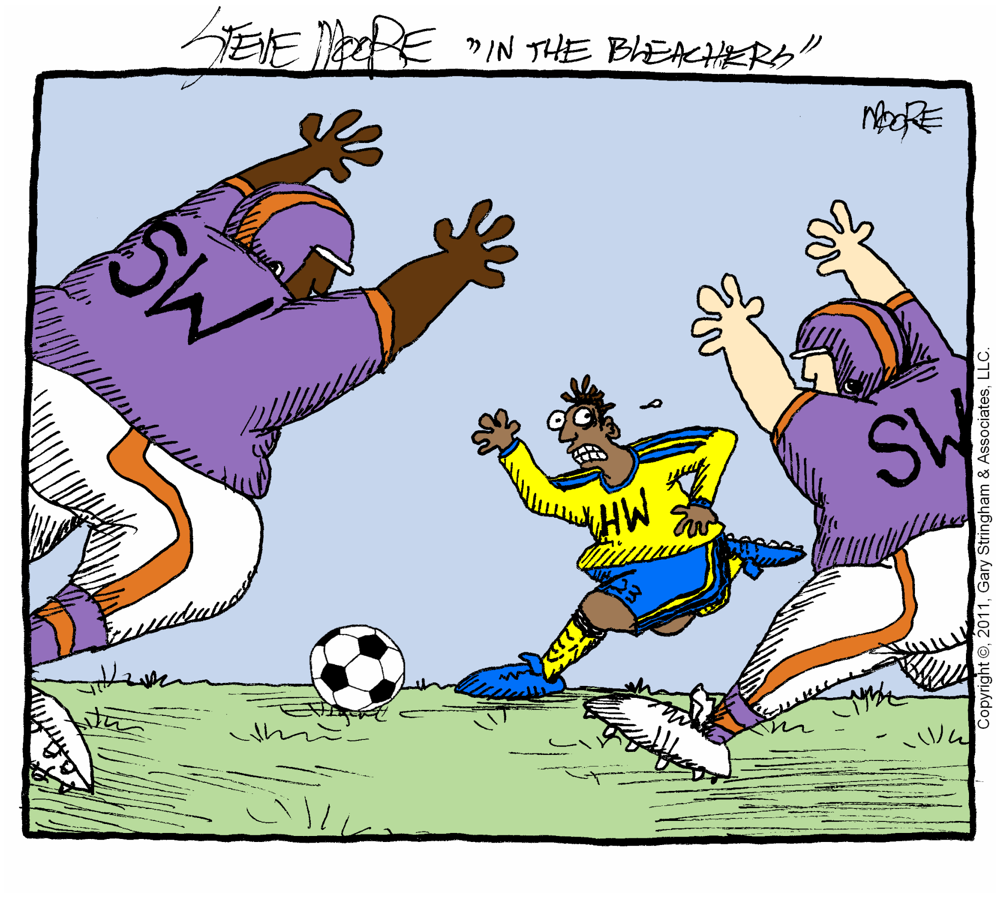
Have All Three

- Faster
- Better
- Cheaper

Fake News: Pick any two

Seven principles → Faster, Better, **&** Cheaper → Greater Profitability

Football or Football?

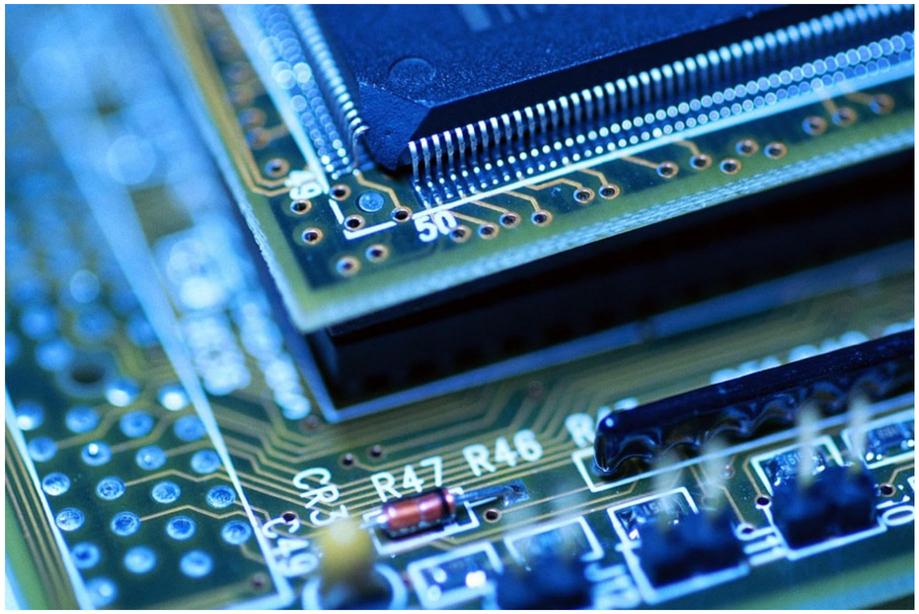

STEVE MOORE "IN THE BLEACHERS"
MOORE
SW
SW
HW
23
Copyright ©, 2011, Gary Stringham & Associates, LLC.

Principle #1 – Collaborate on the Design

- Fake News: “We don’t need the software team’s input”
- Requires pro-active participation on both sides
- Collaborate early
- An important collaborative tool: documentation
- Collaboration is the foundation of all other principles

#1 – Lessons

- Seventh Time’s a Charm – Solve problems using a team of hardware & software engineers
- Permission to Drop – If hardware design request is risky, consult software engineers for alternatives



Principle #2 – Set and Adhere to Standards

- Formalize internal standards
- Stay true to industry standards
- Fake News: Customized standards

#2 – Lessons

- How to Ack an Interrupt – Design all interrupts where a write of a 1 acks
- Insufficient Documentation – Establish a standard and have a review process for all documents

Principle #3 – Balance the Load

- Fake News: “Let’s just let software handle this”
- HW & SW each have strengths and weaknesses
- Different products require different balance settings
- Proper balance improves performance, stability, and quality

#3 – Lessons

- Go, Go, Go – For bits that invoke a hardware task, hardware must clear it after software sets it
- Wait for How Long? – Provide indication to software that hardware is ready

Principle #4 – Design for Compatibility

- Fake News: “Software can easily accommodate this change”
- Permits any SW version to be paired up with any HW version (ideally)
- Designed for all but with only desired features enabled
- New versions paired with old versions won’t break
- Improves future productivity

#4 – Lessons

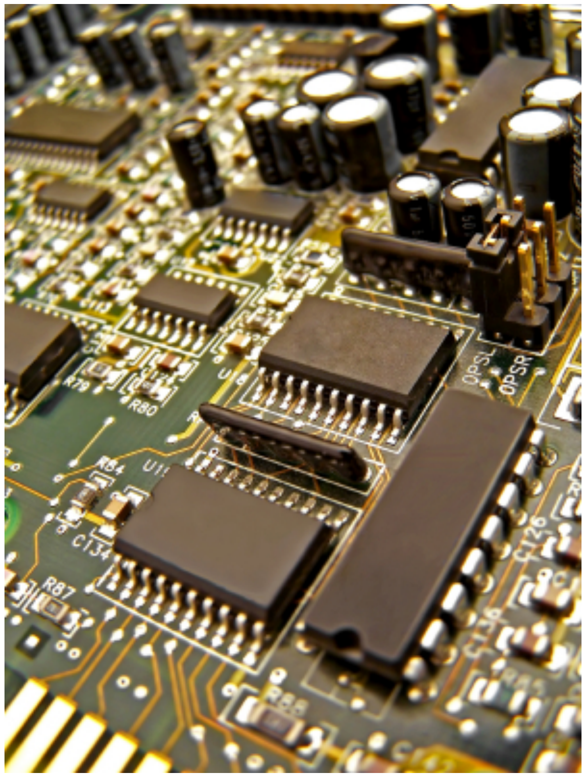
- Remove the Unused Signal – Don’t remove functionality because one target product does not need it; other products might

Principle #5 – Anticipate the Impacts

- Fake News: “Here’s a cool, new feature”
- Anticipate! Don’t just understand or prepare
- Don’t just avoid negative impact – bring in positive impact
- Collaborate to understand the impact

#5 – Lessons

- New Chip Quickly – Limit changes to quickly produce a new version

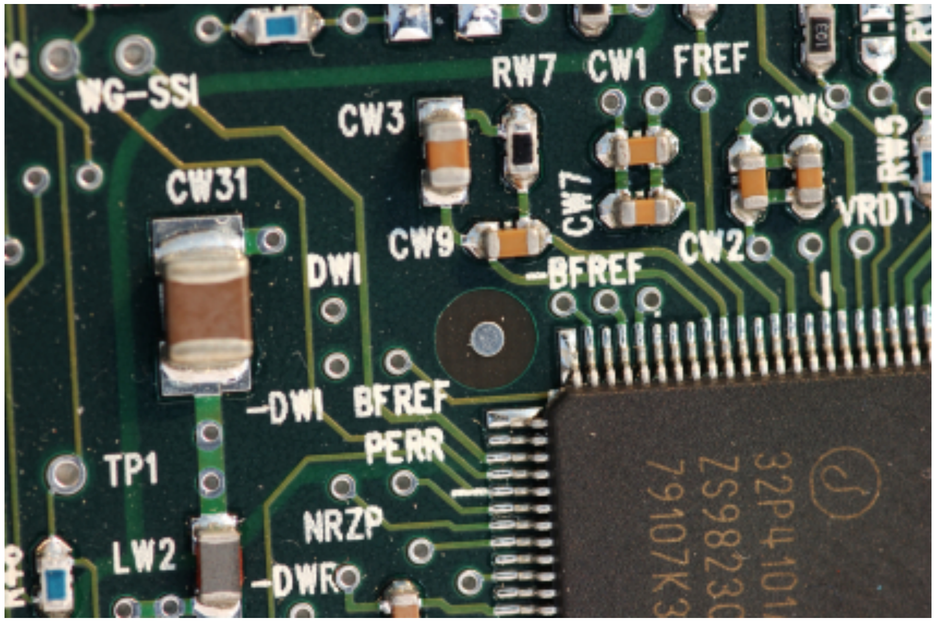


Principle #6 – Design for Contingencies

- Problems will come up – prepare for them
- Design in extra test and debug hooks, even if unlikely to be used
- Add extra registers for internal peek and poke access
- The few hooks that are used makes all hooks worth it
- Fake News: Remove test hooks

#6 – Lessons

- Artificial Signal Generator – Include a hook to simulate external signals
- State of the State Machine – Provide a register showing current state



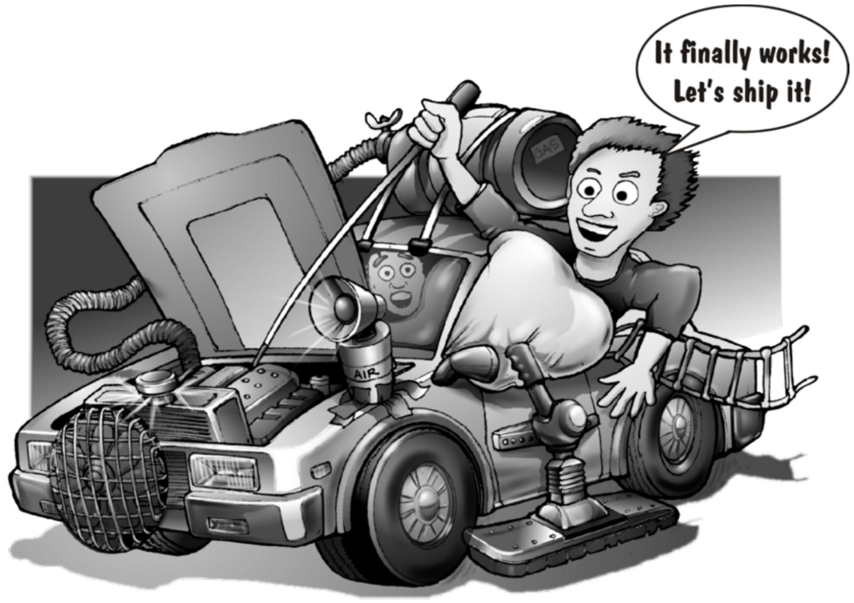
Principle #7 – Plan Ahead

- Fake News: “There is never enough time to do it right, but there is always time to do it again.”
- Good decisions today pay off in the future without sacrificing the current product
- Put in a framework that will allow growth
- Modularity, abstraction, and reuse
- Do things right the first time

#7 – Lessons

- This Version Has It All – Design the block with a superset of functionality

Successful Application of these Principles



Though cobbled together on the inside…

It finally works!
Let's ship it!
HW FW 1

it was a fully-functional and shippable product

The Seven Principles of Successful Hardware/Software Interface Design

1. Collaborate on the Design
2. Set and Adhere to Standards
3. Balance the Load
4. Design for Compatibility
5. Anticipate the Impacts
6. Design for Contingencies
7. Plan Ahead

Newnes
INCLUDES FREE NEWNES ONLINE MEMBERSHIP
HARDWARE/FIRMWARE INTERFACE DESIGN
BEST PRACTICES FOR IMPROVING EMBEDDED SYSTEMS DEVELOPMENT
• Reduce product development delays of ASICs, ASSPs, SoCs, and FPGAs
• Principles and best practices taught and illustrated
• Provides answers to common problems and pitfalls, saving time and effort
Gary Stringham

Rich Weber Presents

FROM THE RTL TEAM'S PERSPECTIVE

Shiny, New Prototype

Software Bringing up Drivers

Didn't Work

Two Weeks of Imprisonment

Time, Money, and Market Share

Getting products out sooner is better than later.

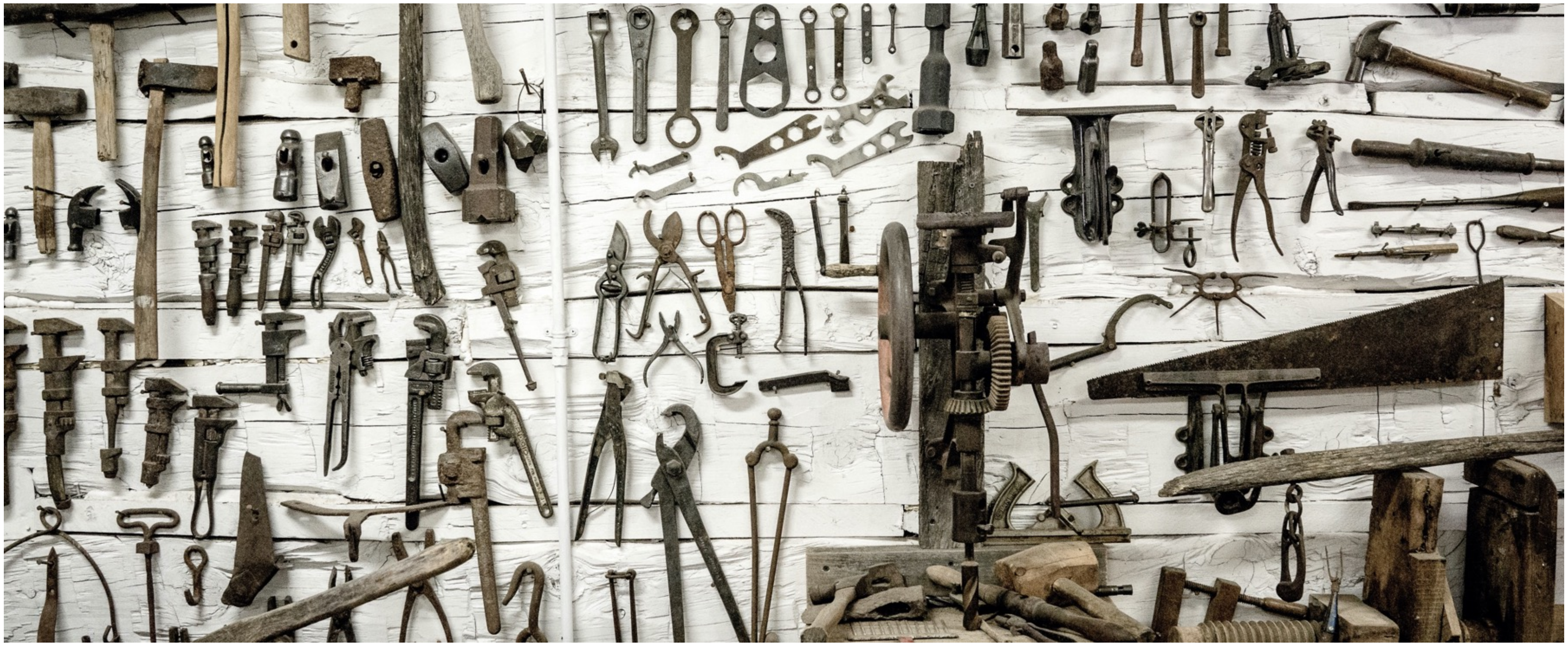
DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION 2020 UNITED STATES

Why Was I Imprisoned?

accellera SYSTEMS INITIATIVE

A Typo

We Had Home Grown Tools.



To Stop Manual Entry

But …

Software Team Didn't Use It

Need the right tool to produce the right outputs.

Homebrew Automation Didn't Solve Our Problem.

You'd Think We'd Learn?

Chip Design in the Wild

Software Wanted More Drivers

Drivers Couldn't Be Created

It Required A Respin

Time, Money, and Market Share

We Finally Learned

Single Executable Source for Address Map

AUTOMATION
TOOLS

Talk with the Software Team

Use Structures That Help

Use Arrays

Will The Industry's Standards Save Us?

Standards Involving HSI

Accellera SystemRDL 2.0

Allows RTL designers to capture RTL and software semantics

IEEE 1685-2014 IP-XACT

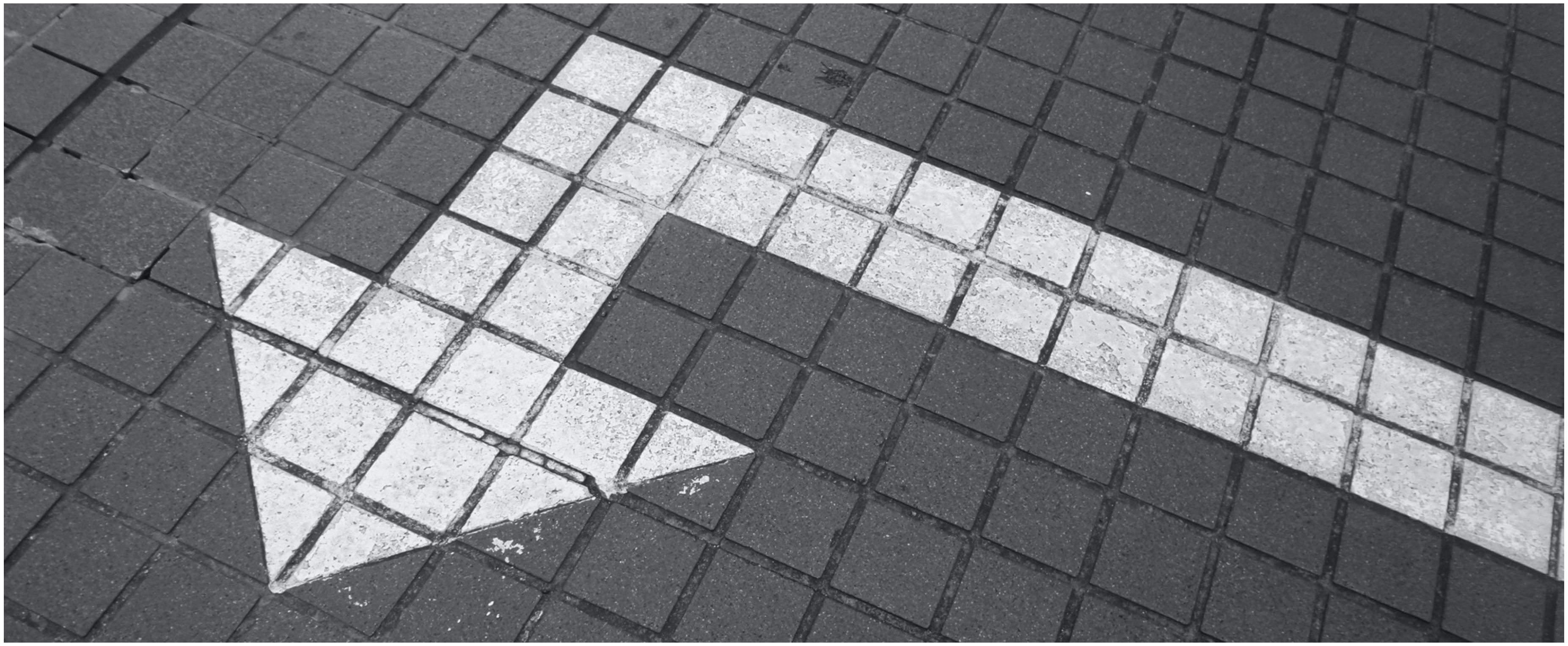
Passes Address Map information between IP providers and design teams

IEEE 1800.2-2017 UVM

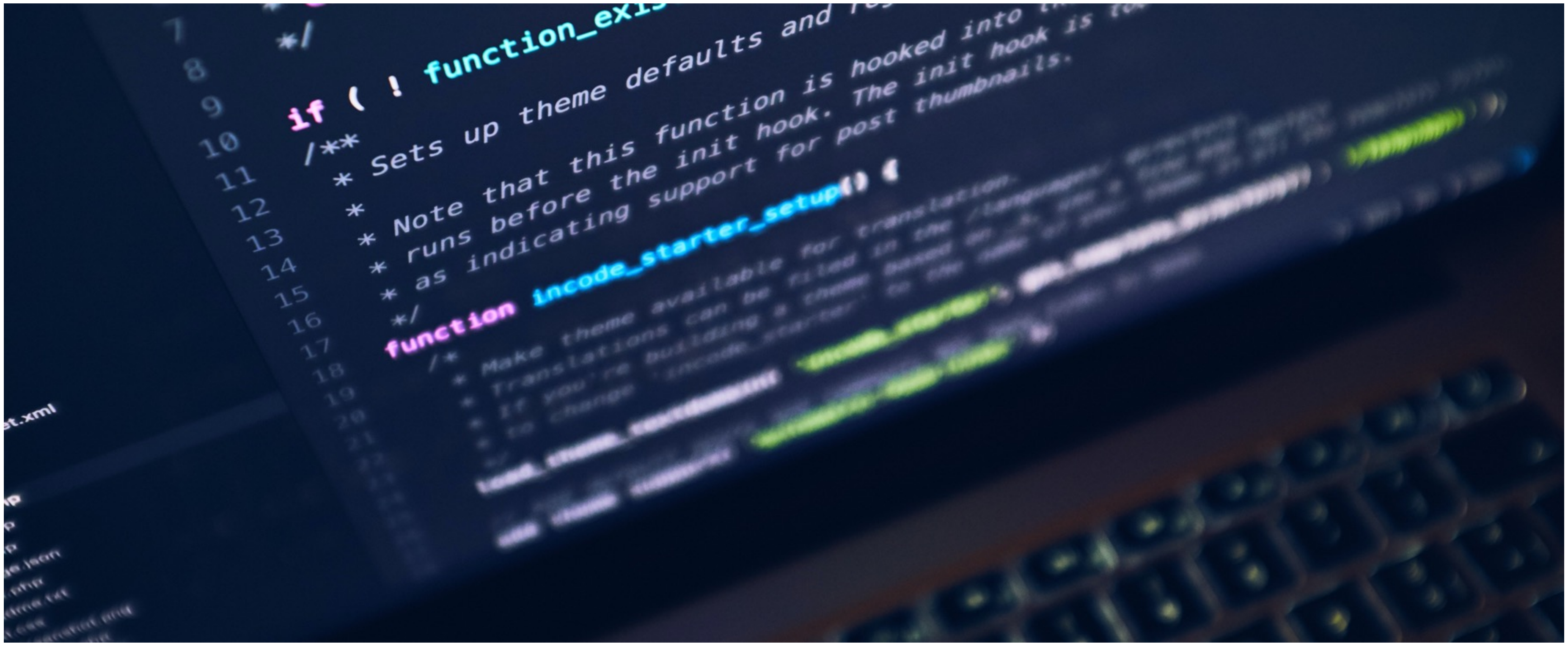
Class support for Address Map structures to benefit the verification team

Where Standards Fall Short

Avoid the Anti-Standard



What About Scripts?



```
*/
if ( ! function_exis
/**
 * Sets up theme defaults and
 *
 * Note that this function is hooked into
 * runs before the init hook. The init hook is
 * as indicating support for post thumbnails.
 */
function incode_starter_setup() {
    /*
     * Make theme available for translation.
```

Be Wary of Homebrew Automation

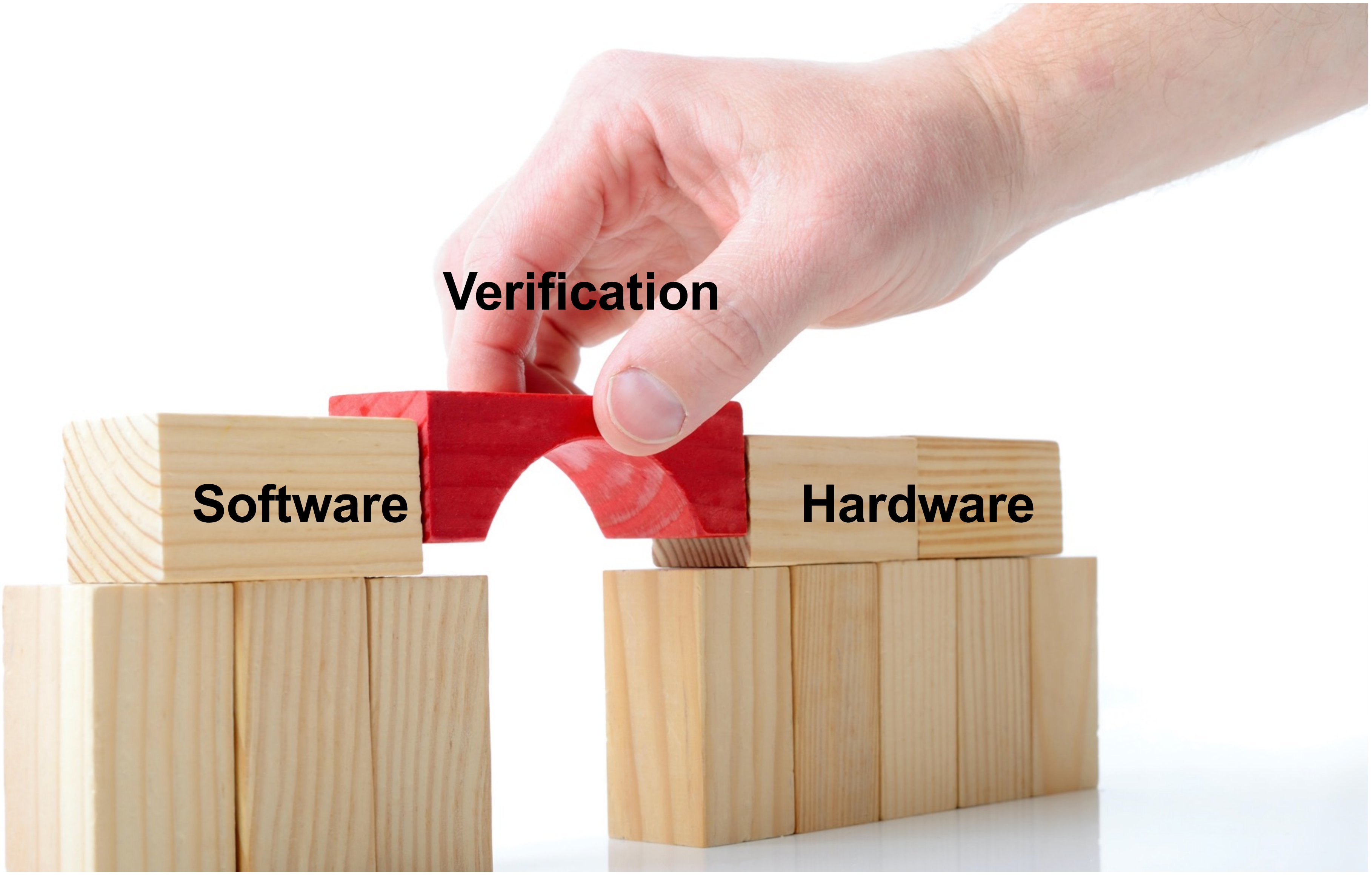
And Open to Commercial Tools

CSRCompiler

- 14 years of Industry Validation
- Designs with over four million registers

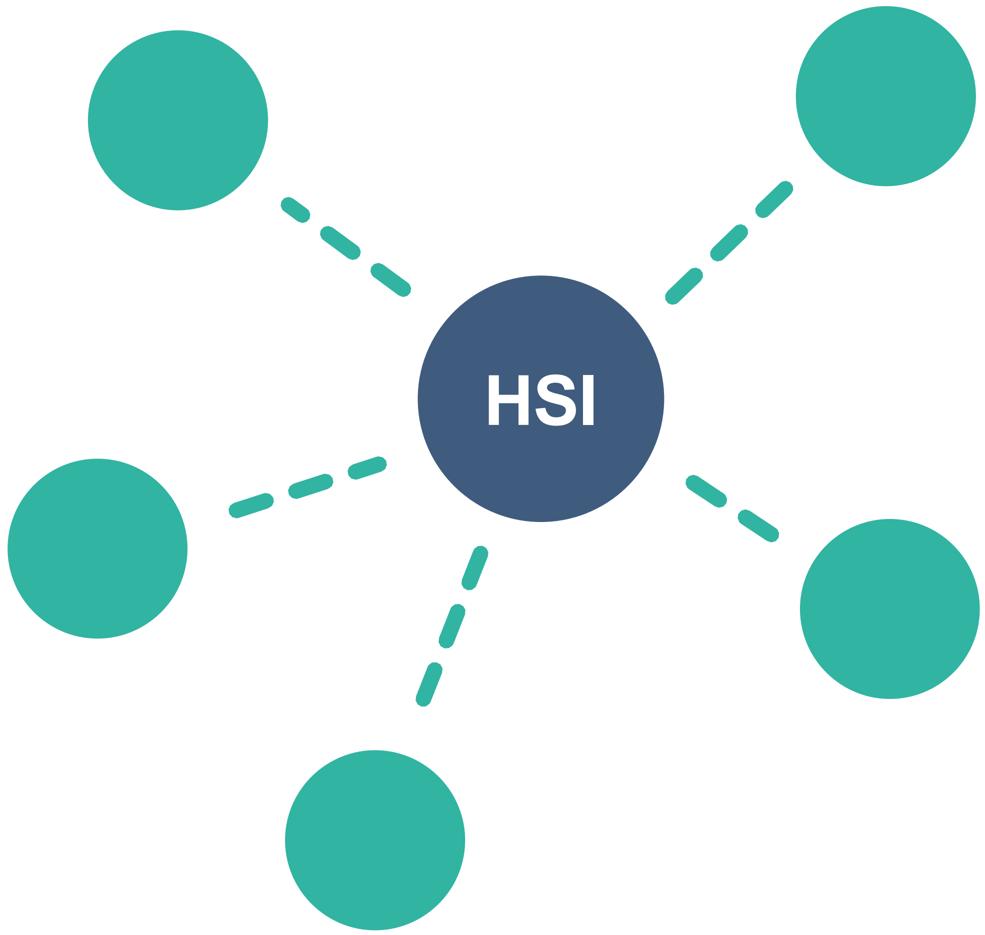
Jamsheed Agahi Presents

FROM THE VERIFICATION TEAM'S PERSPECTIVE


Verification
Software
Hardware

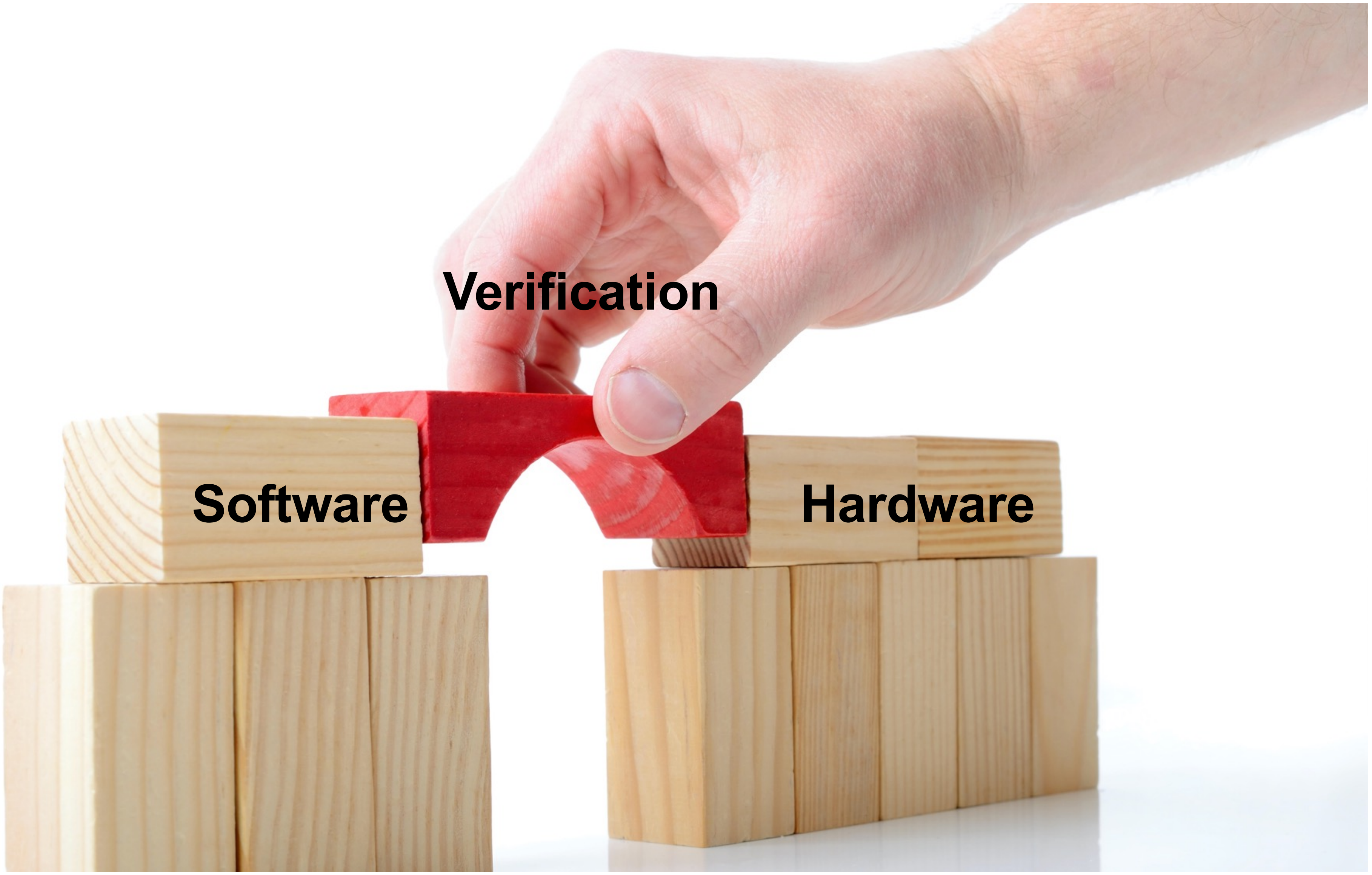
Reduce HSI verification impact on schedule

One HSI,
Multiple Disciplines



Verification Engineer’s Role

- Design the testbench
- Write sequences/tests

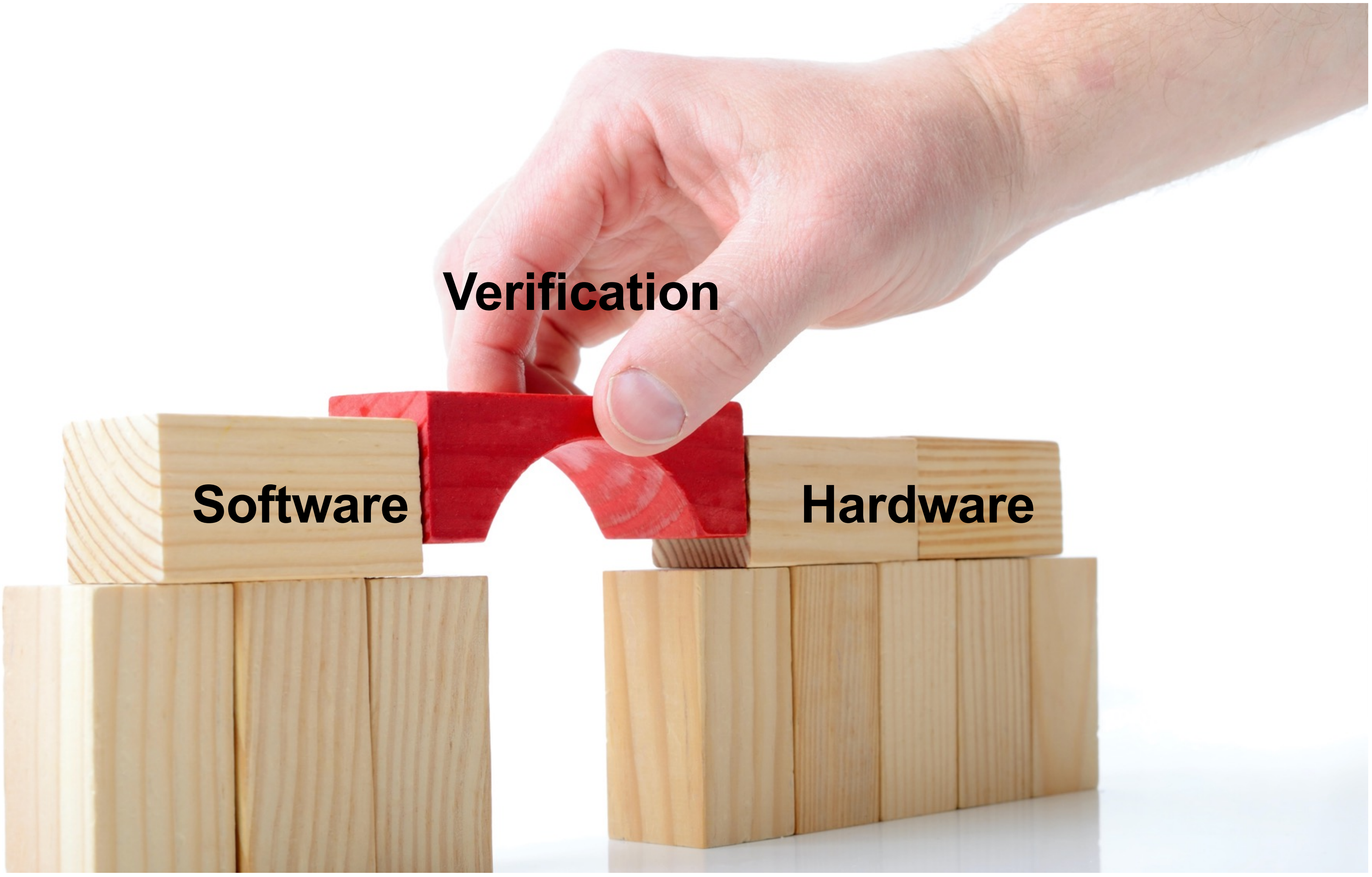

Verification
Software
Hardware

Understand Cost Impact - Software

Understand Cost Impact - Software

Minimize software impact on tape-out

- How software interacts with the HSI
- Discover HSI issues early
- Simplify the HSI

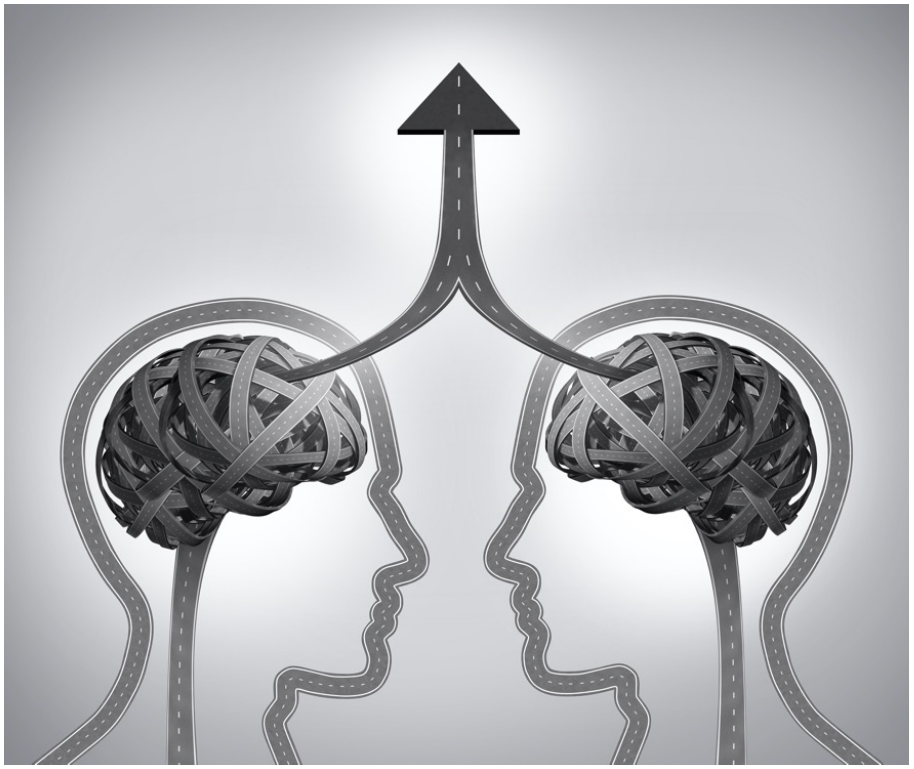

Verification
Software
Hardware

Understand Cost Impact - Verification

Minimize cost and impact on tape-out

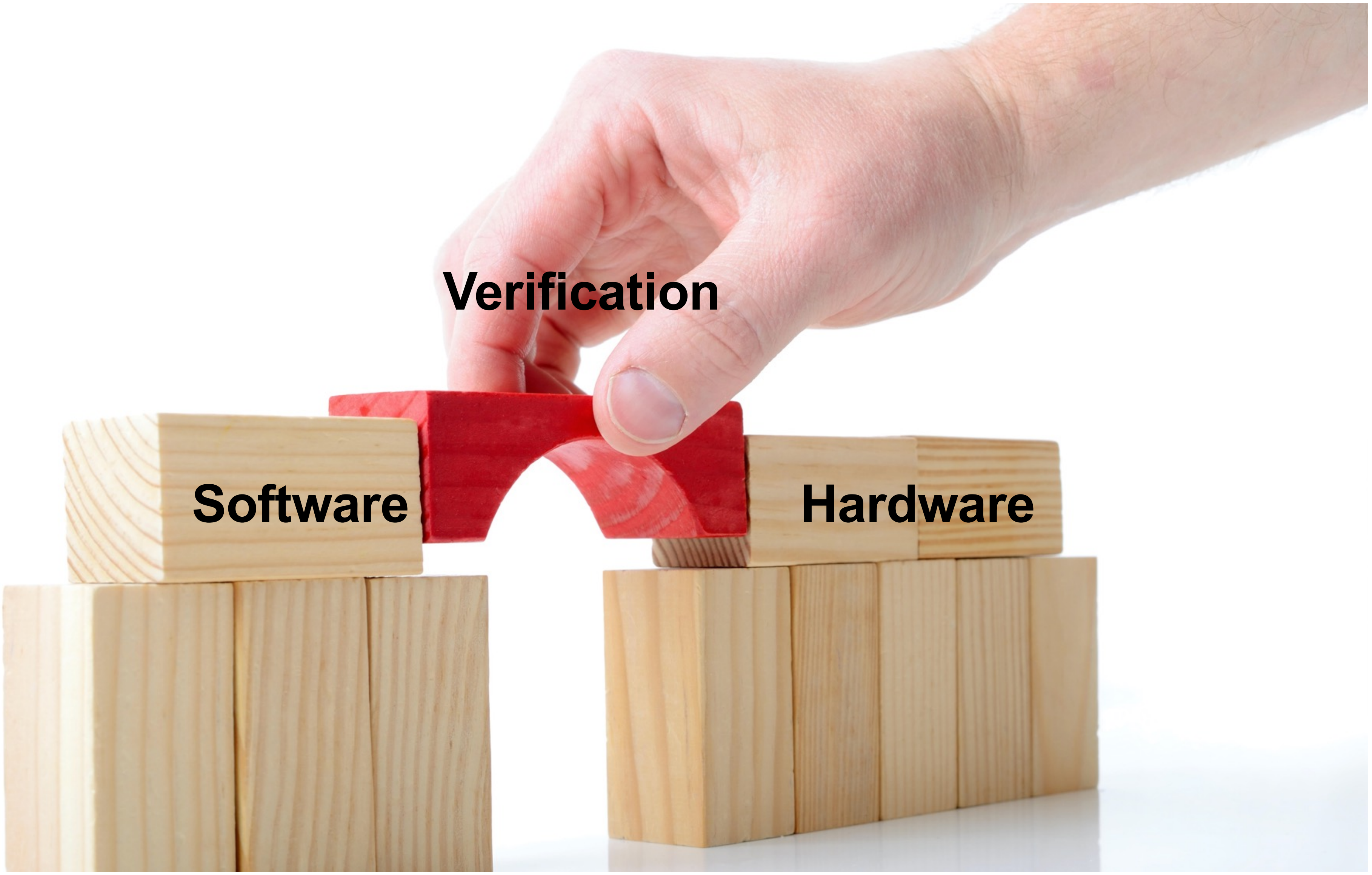
- Adopt a standard methodology – UVM
- Automate generation of register map model
- Automate generation of the HSI documentation
- Focus on verification, not tool development

Be Collaborative



- Participate in architectural discussions
- Understand the bigger picture
- Be aware of product evolution

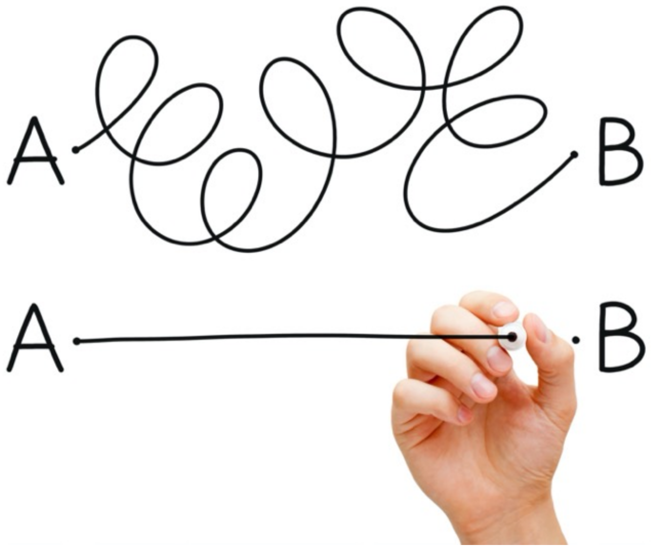
Plan
1 Reuse
2 Modularization
3 Abstraction
4
5


Verification
Software
Hardware

Discuss HSI verification plan with the software team

Be Bold - Request Changes

- Request simplifications to the HSI
- Keep out features not supported by standard verification methodologies (UVM)
- Visibility and debug hooks



Keep Unnecessary Changes Out!

- Register offsets in a block
- Field positions in a register