Applying Agile To Hardware Development

(…We're Not That Different After All!)

Neil Johnson
XtremeEDA
nosnhojn@gmail.com

http://www.AgileSoC.com

About Me

- Neil Johnson
  - 12 years of hardware design and verification
  - Altera, Neterion, Flextronics, Nextwave Wireless
- Principal Consultant XtremeEDA
  - Consulting services
    - Verification experts
  - Clients are any size and many applications
    - Telecom, networking, wireless, computer hardware, etc.
  - We work remotely or onsite as part a client’s team

About Me

- Neil Johnson
  - 12 years of hardware design and verification
  - Altera, Neterion, [illegible], Nextwave Wireless
- Principal Consultant XtremeEDA
  - Consulting services
    - Verification experts
  - Clients are diverse and many applications
    - Telecom, networking, wireless, computer hardware, etc
  - We work remotely or onsite as part of a client team

Huh?

Hardware

Developer?

XtremeEDA

Why Are You Here?

	No	Yes
Learn something from an agile expert	❌	
Become experts in Agile embedded development	❌	

Why Are You Here?

	No	Yes
Learn something from an agile expert	✗	
Become experts in Agile embedded development	✗	
An overview of hardware process and the challenges we face (Part I)		✓
See how the principles of the agile manifesto apply to hardware development (Part II)		✓
How hardware teams can get started with agile and how you can help (Part III)		✓

Part I

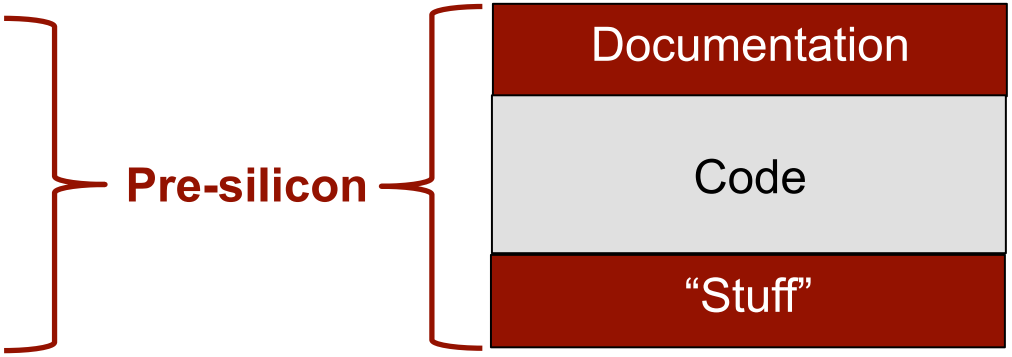
What Are The Strange Hardware People Doing?

What Do I Mean By Hardware

- ASIC
  - Application Specific Integrated Circuit
  - Static structure
  - Digital or mixed signal
  - High NRE/Low cost
- FPGA
  - Field Programmable Gate Array
  - Reprogrammable structure
  - Primarily digital
  - No NRE/High cost
- SoC
  - Either of the above + embedded processor(s) + software

SoC Development Basics

- Typical SoC design flow
  - Specification
  - Design
  - Verification
  - Physical design



Pre-silicon

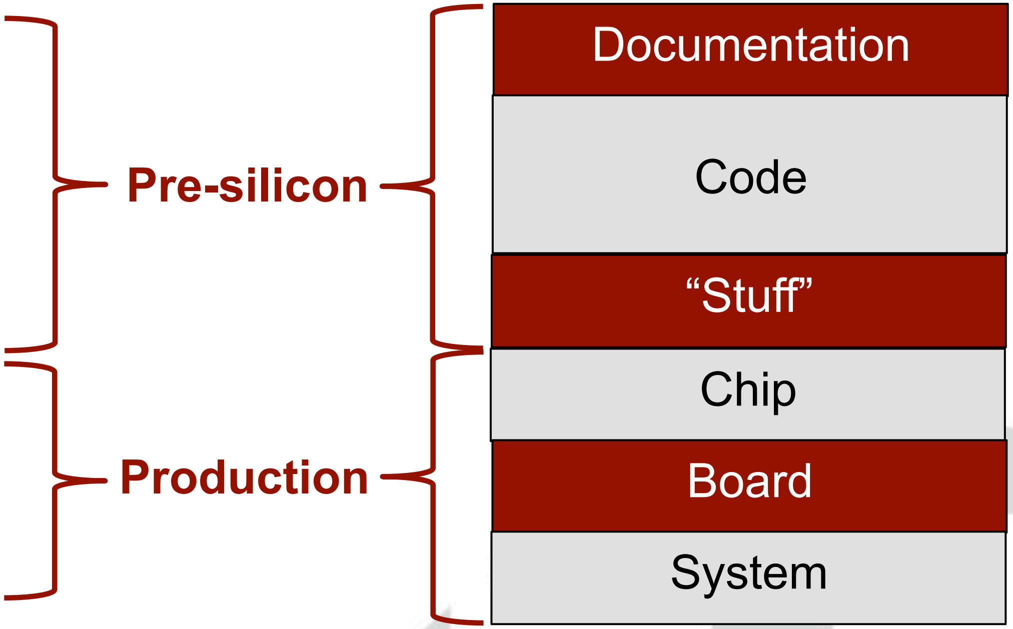
Documentation

Code

“Stuff”

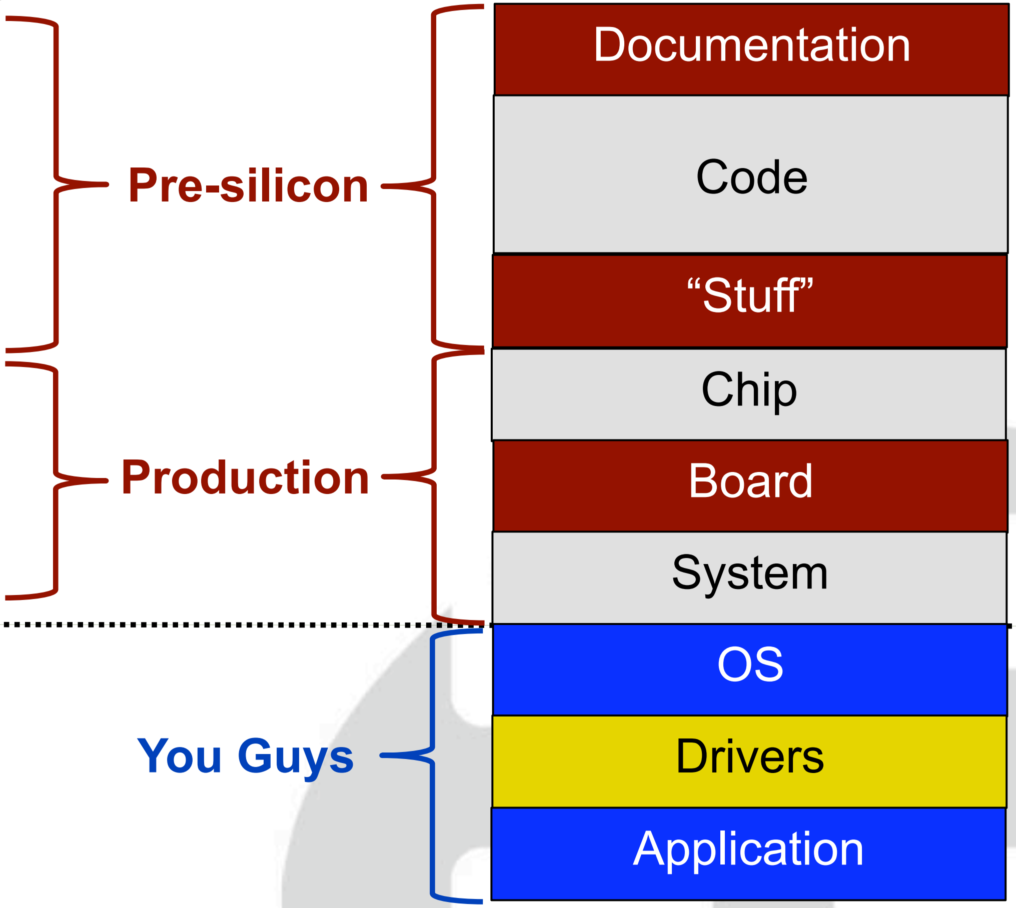
SoC Development Basics

- Typical SoC design flow
  - Specification
  - Design
  - Verification
  - Physical design
  - Fabrication
  - Validation
  - Integration



SoC Development Basics

- Typical SoC design flow
  - Specification
  - Design
  - Verification
  - Physical design
  - Fabrication
  - Validation
  - Integration

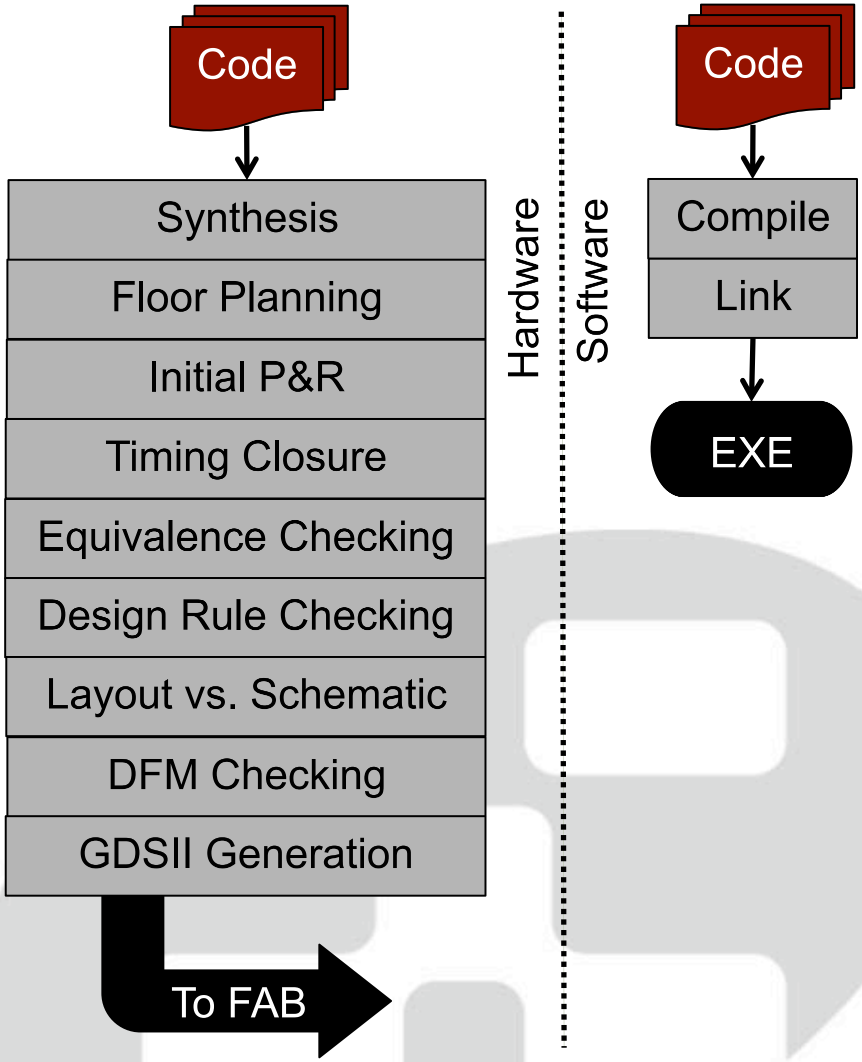


Common Industry Challenges

- Tape-out… aka: the big bang
  - NRE >$1million
- Cost of first silicon > $10Million
- We like to be careful!
- Tape-out stress is very high

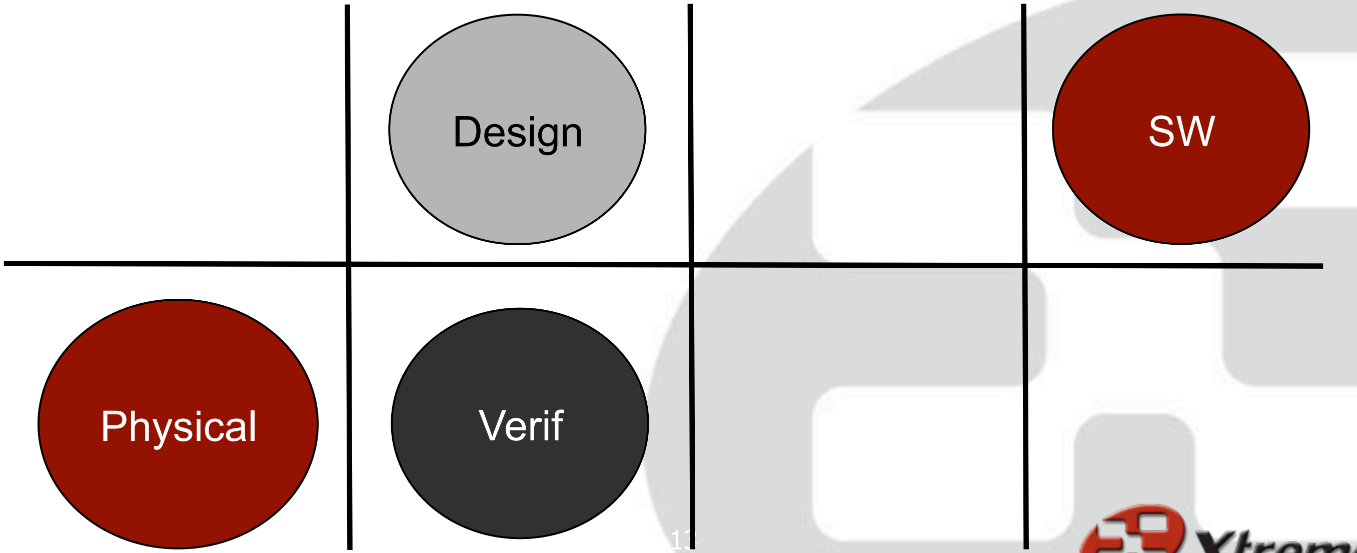
Common Technical Challenges

- Optimization
  - Size, speed, power consumption, target technology and time-to-market
- We depend on tools very, very heavily
  - Physical design is pretty complicated relative to a software build



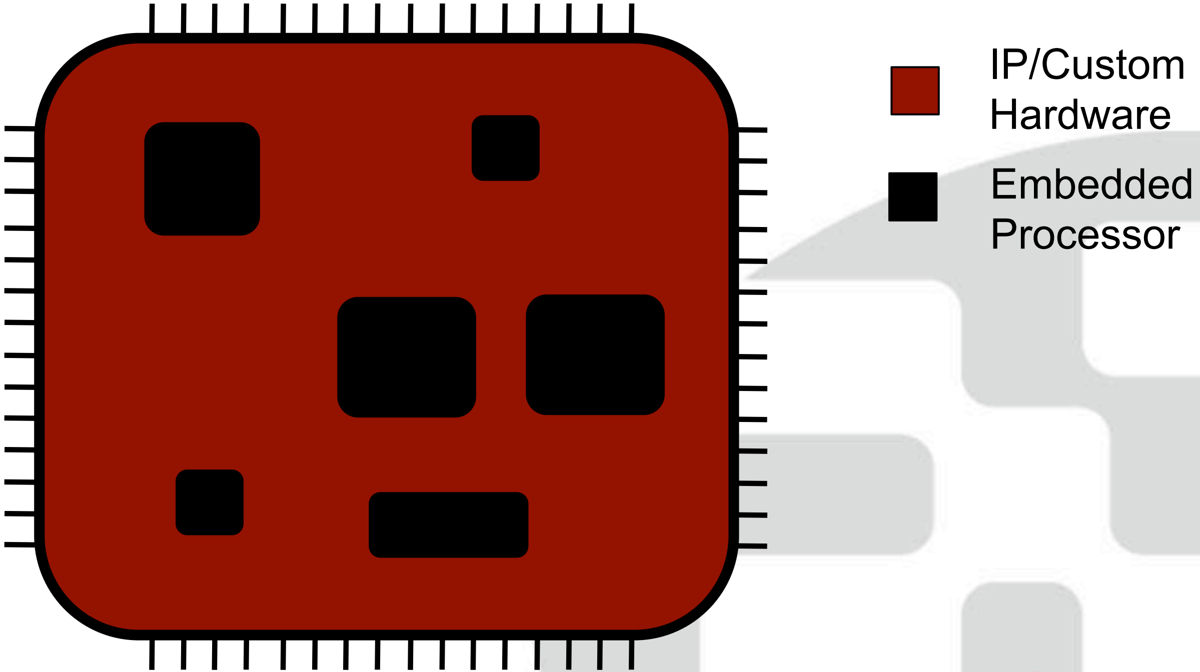
Common Organizational Challenges

- Organized primarily by function
  - Functional teams act independently
    - Design and verification don’t normally share goals
    - Common for software to be “out of the loop”
  - Physical and/or software teams are involved late
    - “deal with it” instead of working together



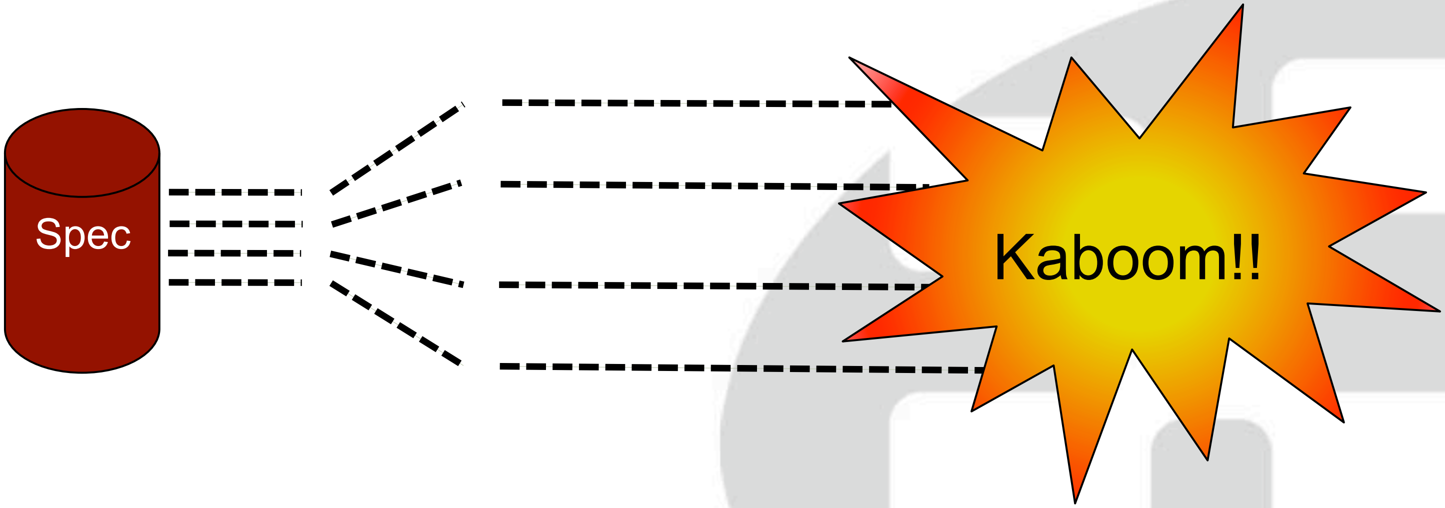
Common Organi-technical(?) Challenges

- The composition of an SoC is changing
  - Software is becoming dominant
  - Tools are being developed to integrate hw/sw
  - Encouragement of real teamwork lags



Big Bang Hardware Development

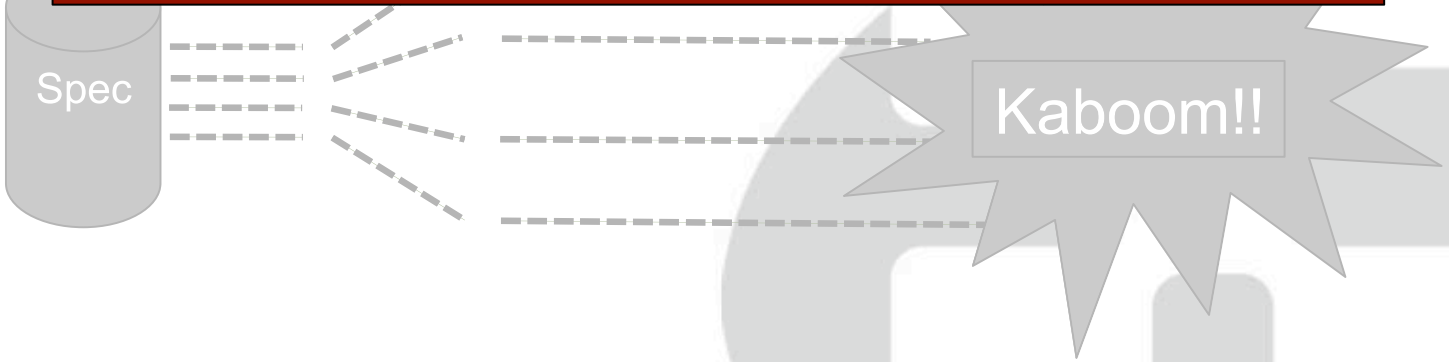
- We do EVERYTHING in parallel with no objective way to measure progress along the way
  - Strict product definition
  - Compartmentalization and vertical team organization
  - Teams quickly diverge; minimal communication
  - Long development times with few checkpoints
  - Subjective development status



Big Bang Hardware Development

- We do EVERYTHING at once with no objective way to measure progress along the way
  - Strict product definition
  - C
  - T
  - L
  - S

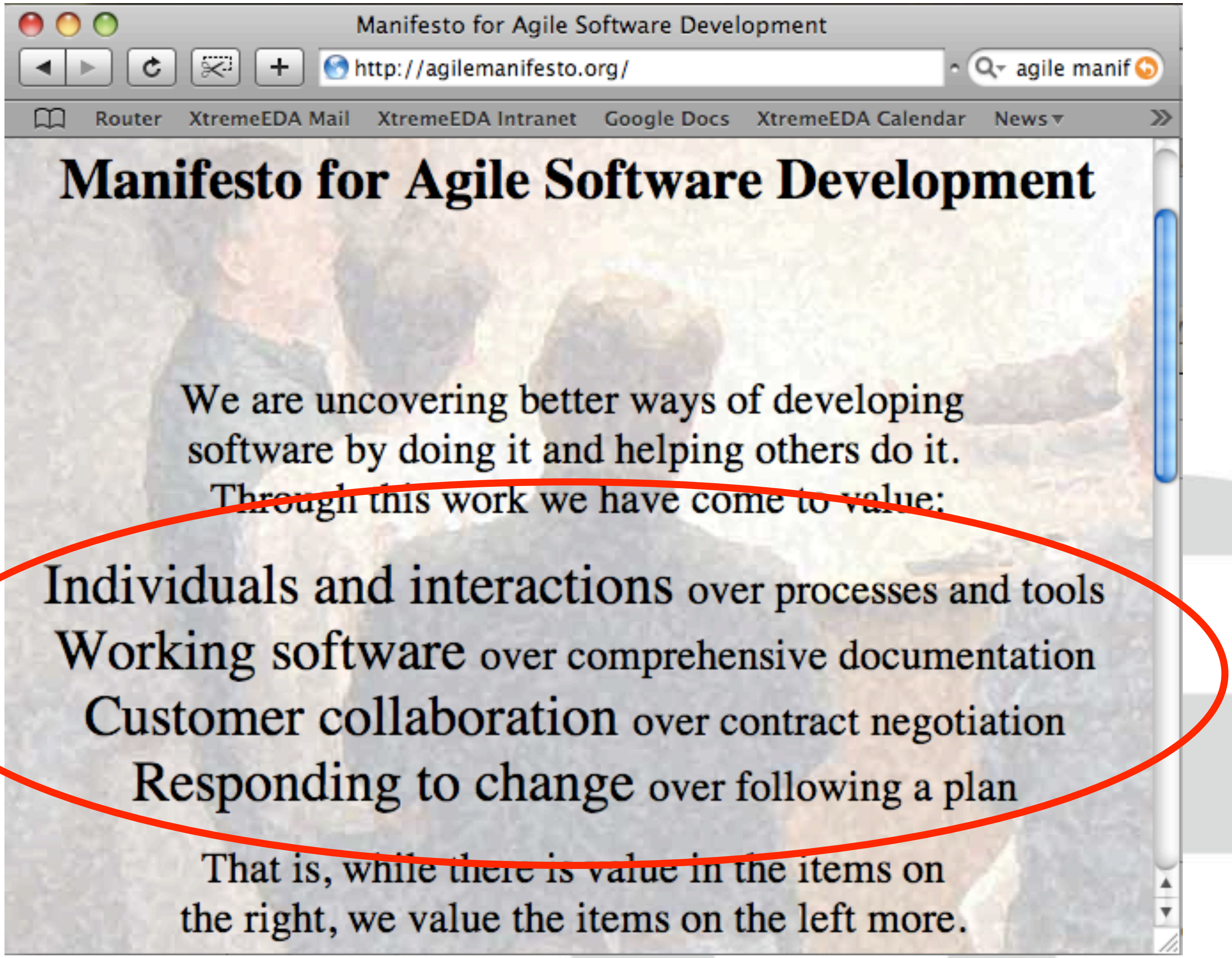
> …if a project managed by a defined process fails, people then assume that the project failed because the defined approach was not adhered to rigorously enough. They conclude that all they need…is increased control and project definition.
>
> Ken Schwaber, Agile Project Management with Scrum



Part II

Taking the Manifesto Where It Wasn't Meant To Go

www.agilemanifesto.org



Manifesto for Agile Software Development

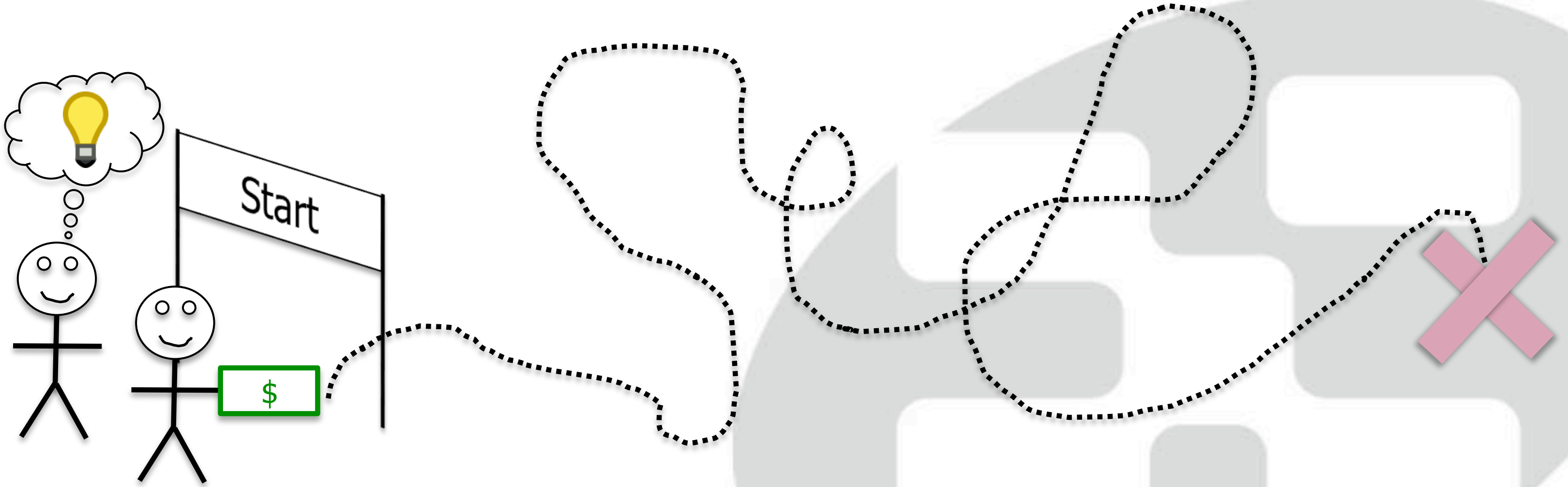
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Customer Collaboration

- Customers can’t tell you exactly what they want
  - Feature creep is actually good!
  - Question what you’re building and why
  - Find the middle ground between acceptance and rejection
- Prioritize requirements



Individuals and Interactions

- Building a cross-functional development team

Individual Ownership Of Individual Tasks

✗

Functional sub-teams
Isolation
Documentation
Cubicles
Code that’s written
Reporting progress

Shared Goals and Cooperative Problem Solving

✓

Functional experts
Pairing
Conversation
Co-located teams
Code that works
Demonstrating progress

Individuals and Interactions

- Visibility and Effective Communication
  - We prefer technical solutions over “people” solutions
    - bug data bases
    - project management sw/spreadsheets

> “I…found that…Purely people factors predict project trajectories quite well, overriding choice of process or technology.”
>
> Alistair Cockburn, *Agile Software Development: The Cooperative Game*

Working Software (Hardware)

- Waterfall Model
  - A sequential process
  - One big bang, production ready release at the end of the project
  - Lessons learned for the next project
  - Task driven development

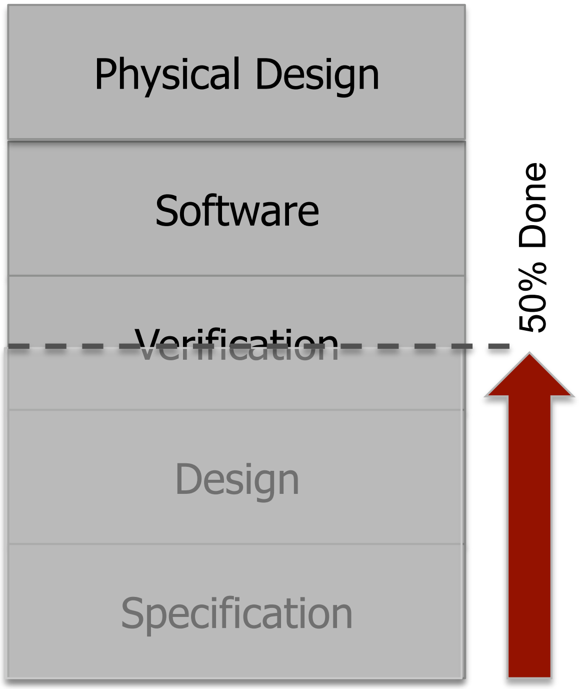
Working Software (Hardware)

- Waterfall Model
  - A sequential process
  - One big bang, production ready release at the end of the project
  - Lessons learned for the next project
  - Task driven development

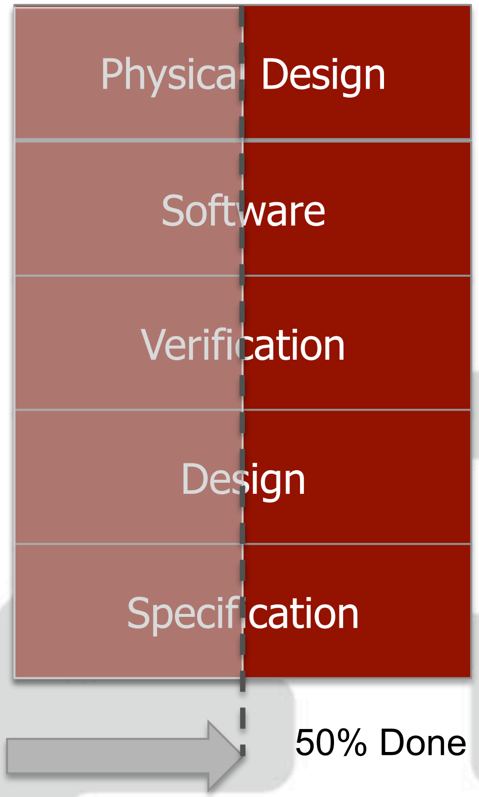
- Agile model
  - An iterative process
  - Many production ready "re-spins" during the project
  - Many opportunities for feedback
  - Deliverables driven development with every iteration

Working Software (Hardware)

- Waterfall Model

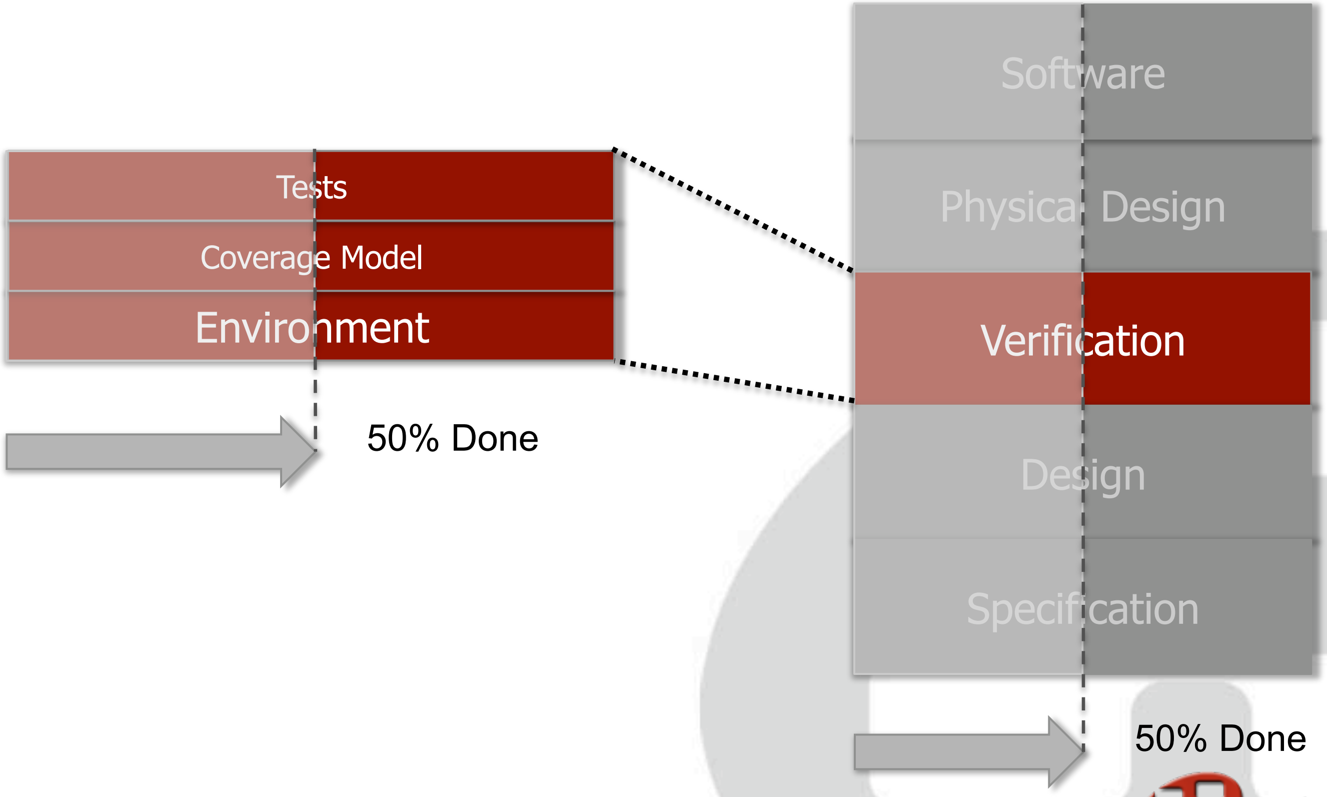


- Agile model



Working Software (Hardware)

- Incremental coverage closure
  - Coverage model grows incrementally with the DUT



Responding to Change

- Regular regression testing
  - Add new features without breaking existing features
  - Start on day 1 to avoid introducing defects
- Continuous integration
  - Small batch integration == quick & easy
  - large batch integration == slow & difficult

The Solution: Agile Development

- Instead of building everything at once, build a product incrementally
  - Everything you would normally do, just do it a little differently a little piece at a time

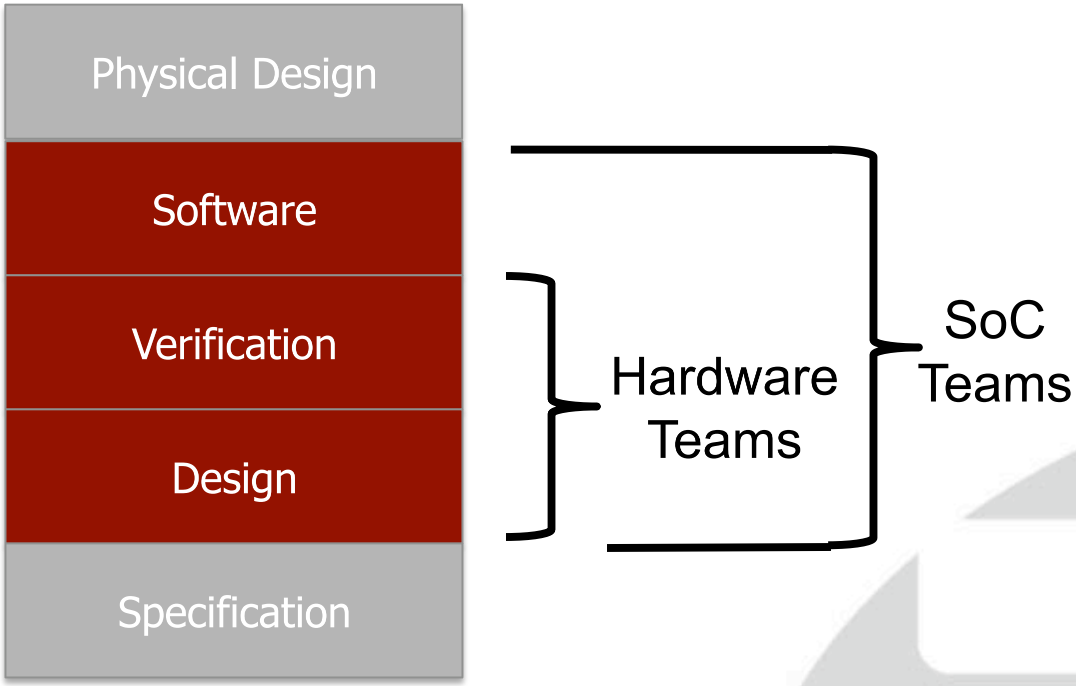
Part III

If it looks like a Duck…
And it quacks like a Duck…

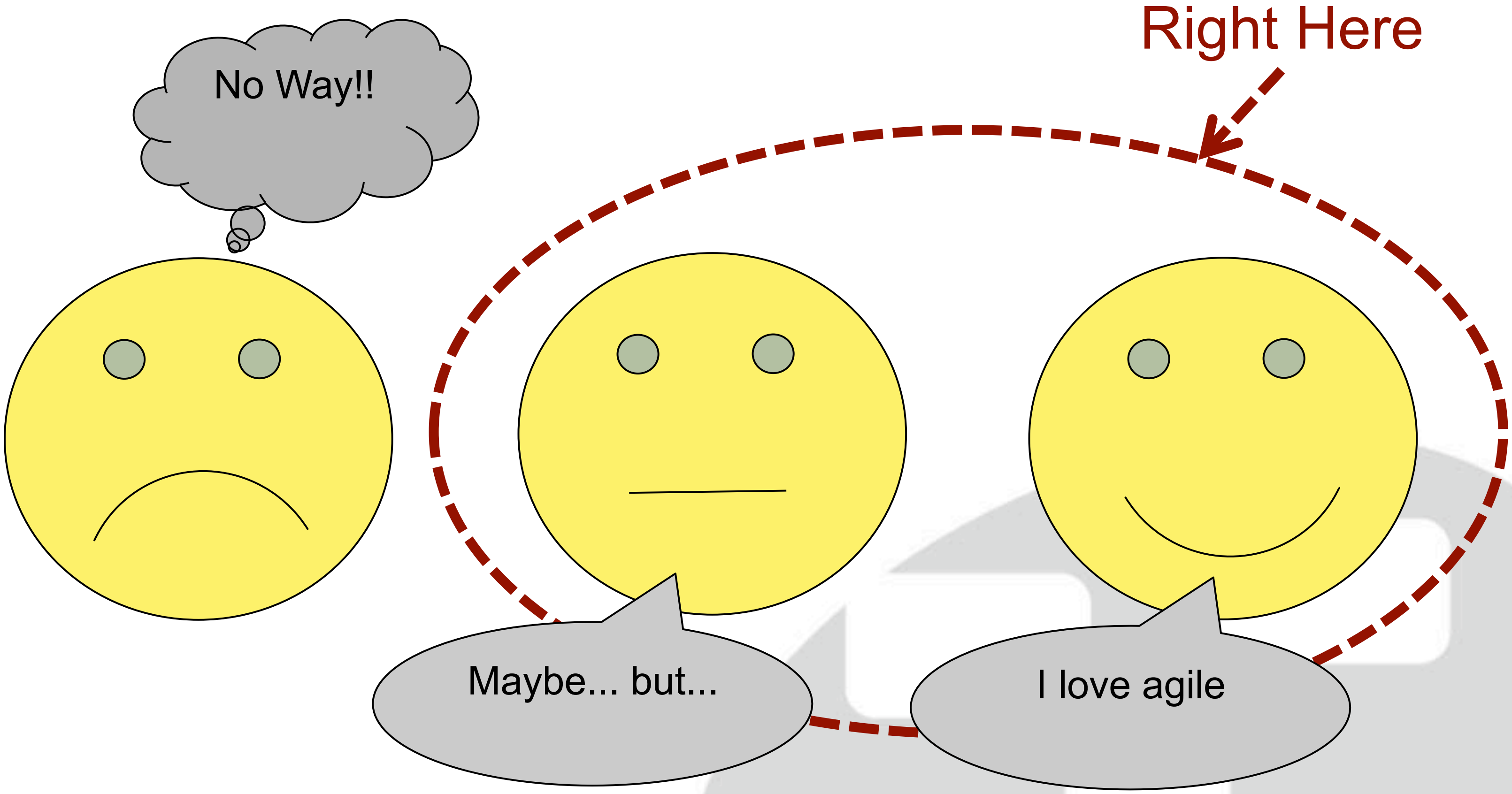
You're better off saying it's a Rabbit.

Where's The Potential?

- Agile model

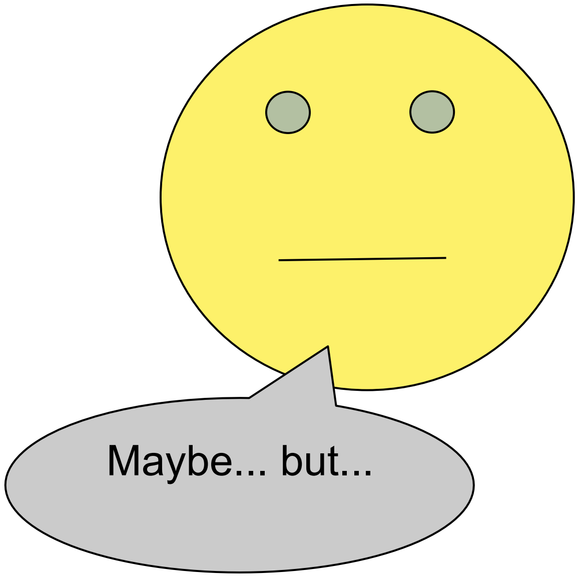


Where's The Potential?

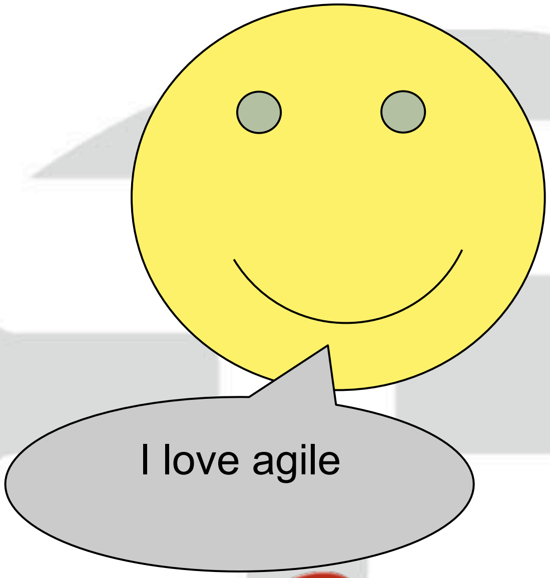


Where's The Potential?

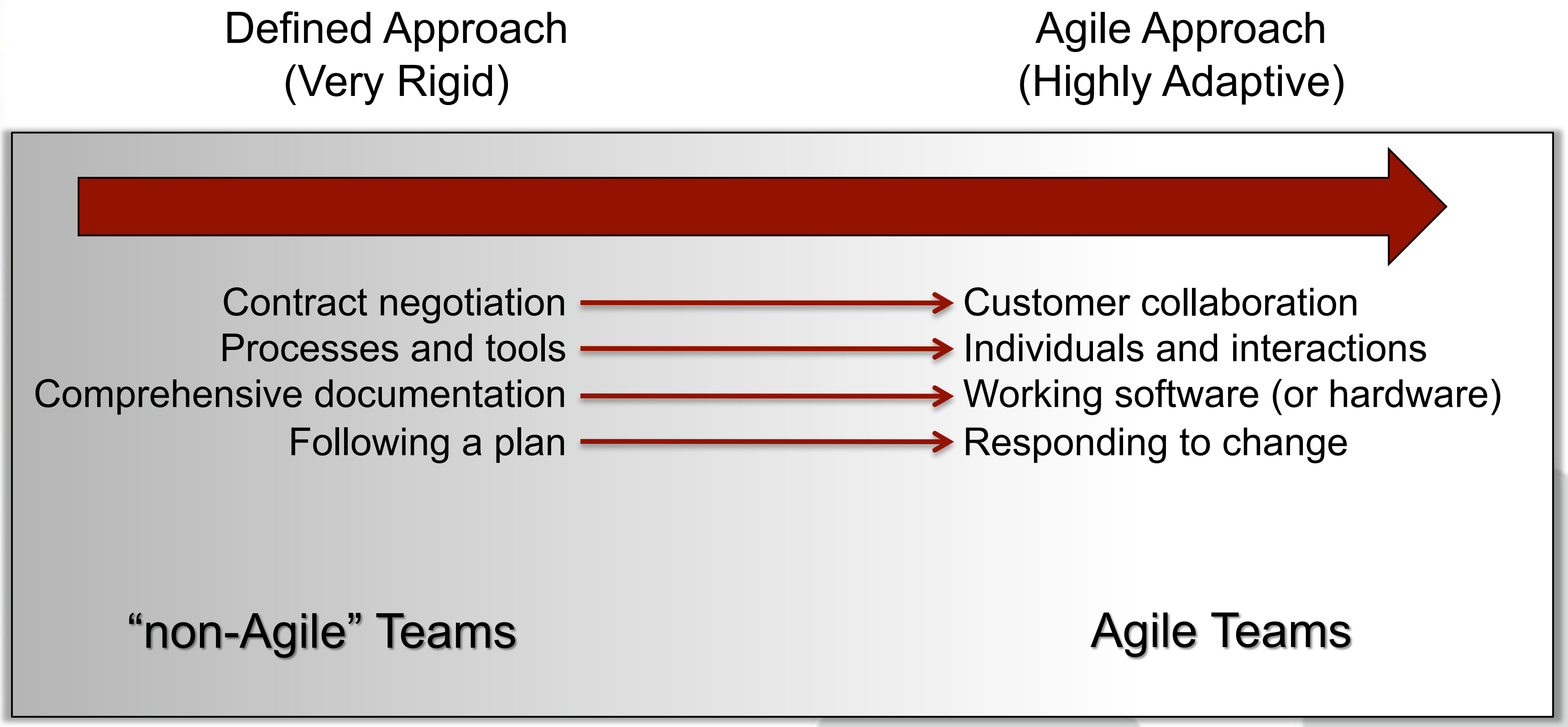
- Deviations from traditional hardware development
  - "Here is a different way to look at the things you already do"



- Agile articles and ideas
  - "Agile software developers have a lot of good ideas. We should use some of them"



Getting Comfortable: www.agilemanifesto.org



Defined Approach
(Very Rigid)

Agile Approach
(Highly Adaptive)

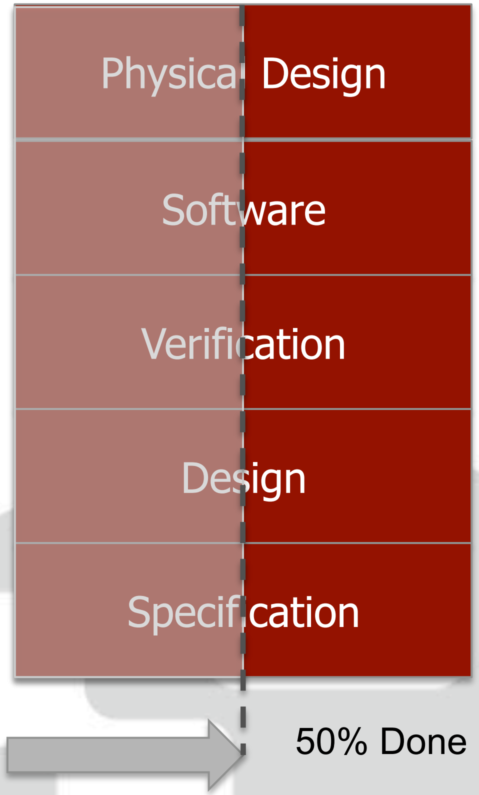
Contract negotiation → Customer collaboration
Processes and tools → Individuals and interactions
Comprehensive documentation → Working software (or hardware)
Following a plan → Responding to change

“non-Agile” Teams

Agile Teams

Where's The Potential?

- **Working hardware over comprehensive documentation**
  - We complete tasks, not features
    - Client experience #1: The Feature-of-the-week
  - We develop everything at once... we can't help it!
    - Client Experience: Just try and deliver something (anything)
    - Exercise: Operation Basic Sanity



The Perfect Place To Start

- Feature-of-the-week
  - “Tell your customer you’re going to give them something that works in a week”
    - Jonathan Rassmuson, *Agile in a Nutshell*, APLN April 2009
  - The Agilesoc Blog: Remote Development And The Feature-of-the-week

The feature-of-the-week is not Agile, it’s frequent delivery... which is hard to argue against.

Feature Of The Week

- Client experience
  - Goal
    - Convince myself that incremental development is possible
  - Situation
    - Functional testing of a sub-system
    - Design was done, test harness was partially complete
  - Planning
    - 4 increments, 1 for each major feature
    - Detailed plan included 1-2 week sub-milestones
    - 2 increments planned in detail

Feature Of The Week

- Client experience - Highlights
  - Planning
    - The planning was different but no convincing was required
  - Increment 1
    - Uh oh… I've committed to delivering something in a week
    - First up: remove everything I don't need

Feature Of The Week

- Client experience – Highlights (con't)
  - Increment 2
    - I was focused and delivering on time
    - Functional milestones allowed me to react to new priorities
  - Increment 3
    - Functioning code was great for gaining confidence and/or being corrected
    - I wasn't so concerned with building infrastructure
  - Summary
    - ✓ Convince myself agile can work

Client Experience: Just Try And Deliver

- Situation
  - Deliver an internal IP block
  - Project well behind schedule
  - Just coding the design and test environment would take us beyond our scheduled delivery date
- Good time to introduce agile as an alternative to tradition
  - dealing with agile skeptics
  - I wanted to make sure that different still familiar so I didn't scare anyone away

Client Experience: Just Try And Deliver

- Recommendations
  - Prioritize a bypass solution to be used in case of emergency
  - Incrementally deliver the rest as a series of threads
  - Change the order in which we do things

Client Experience: Just Try And Deliver

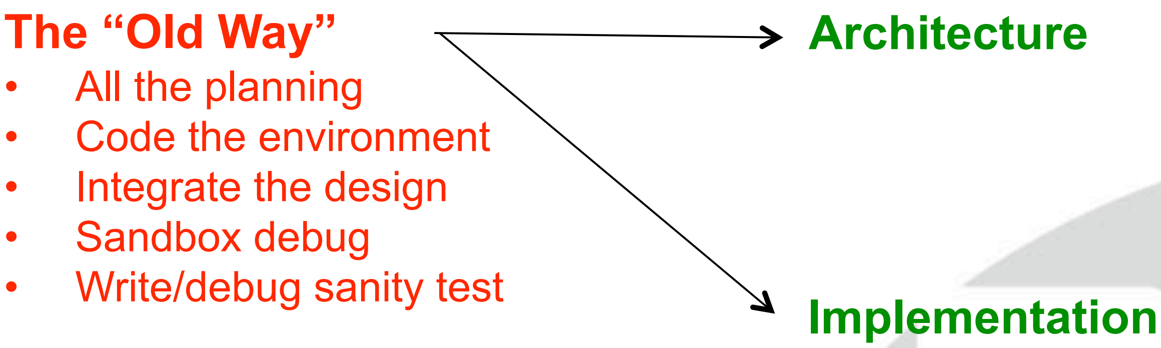
- Recommendations
  - Prioritize a bypass solution to be used in case of emergency
  - Incrementally deliver the rest as a series of threads
  - Change the order in which we do things

The "Old Way"

- All the planning
- Code the environment
- Integrate the design
- Sandbox debug
- Write/debug sanity test

Client Experience: Just Try And Deliver

- Recommendations
  - Prioritize a bypass solution to be used in case of emergency
  - Incrementally deliver the rest as a series of threads
  - Change the order in which we do things



The "Old Way"

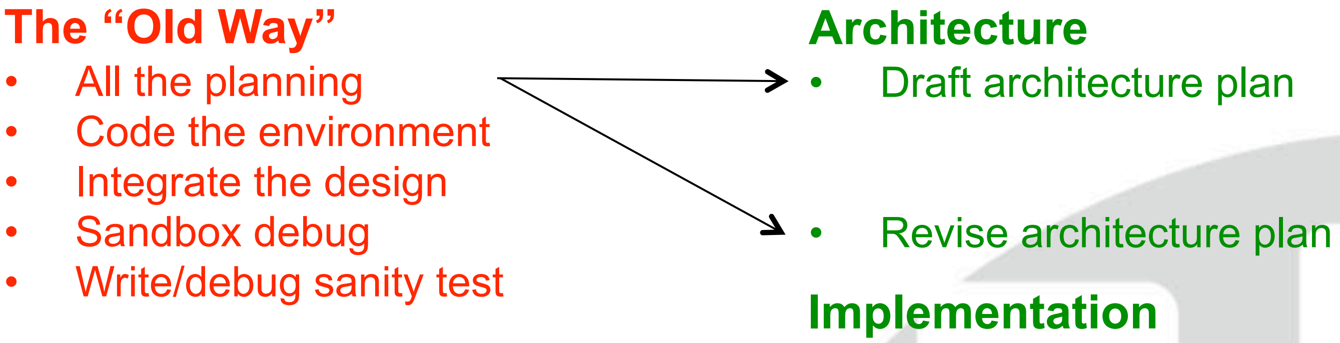
- All the planning
- Code the environment
- Integrate the design
- Sandbox debug
- Write/debug sanity test

Architecture

Implementation

Client Experience: Just Try And Deliver

- Recommendations
  - Prioritize a bypass solution to be used in case of emergency
  - Incrementally deliver the rest as a series of threads
  - Change the order in which we do things



The "Old Way"

- All the planning
- Code the environment
- Integrate the design
- Sandbox debug
- Write/debug sanity test

Architecture

- Draft architecture plan

- Revise architecture plan

Implementation

Client Experience: Just Try And Deliver

- Recommendations
  - Prioritize a bypass solution to be used in case of emergency
  - Incrementally deliver the rest as a series of threads
  - Change the order in which we do things

The "Old Way"

- All the planning
- Code the environment
- Integrate the design
- Sandbox debug
- Write/debug sanity test

Architecture

- Draft architecture plan
- Skeleton test environment
- Revise architecture plan

Implementation

Client Experience: Just Try And Deliver

- Recommendations
  - Prioritize a bypass solution to be used in case of emergency
  - Incrementally deliver the rest as a series of threads
  - Change the order in which we do things

The “Old Way”

- All the planning
- Code the environment
- Integrate the design →
- Sandbox debug
- Write/debug sanity test

Architecture

- Draft architecture plan
- Skeleton test environment
- Integrate design
- Revise architecture plan

Implementation

Client Experience: Just Try And Deliver

- Recommendations
  - Prioritize a bypass solution to be used in case of emergency
  - Incrementally deliver the rest as a series of threads
  - Change the order in which we do things

Operation Basic Sanity: A Faster Way To Sane Hardware

Architecture

- Draft architecture plan
- Skeleton test environment
- Integrate design
- Revise architecture plan

Implementation

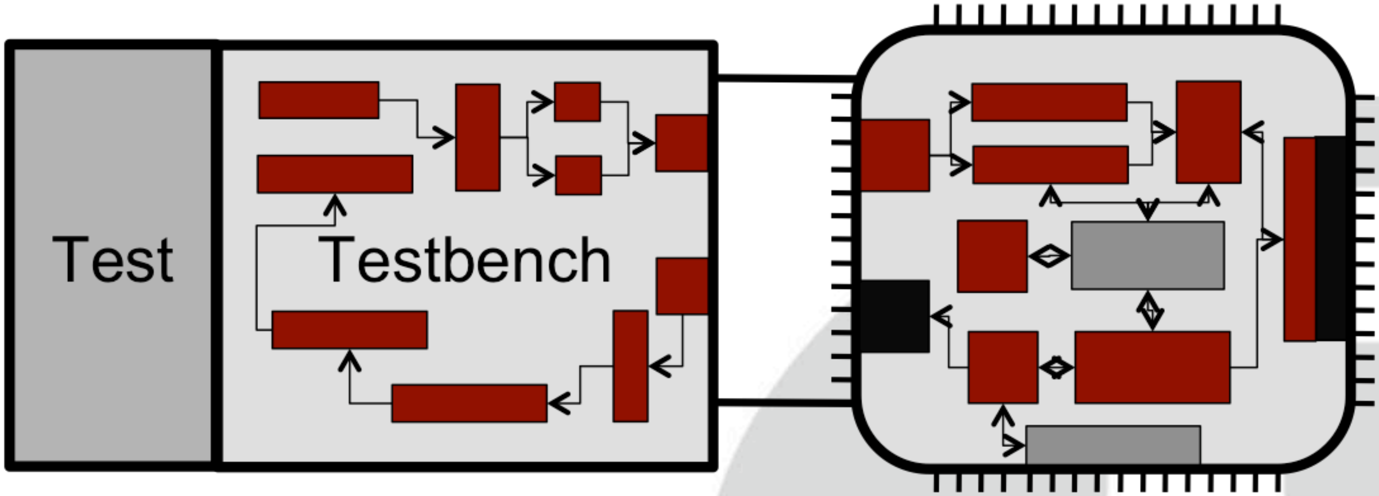
- Draft implementation plan
- Write the sanity test
- Code the required sanity design/test environment
- Debug the sanity test
- Revise the implementation plan

Exercise: Operation Basic Sanity

- Help teams plan a first product increment

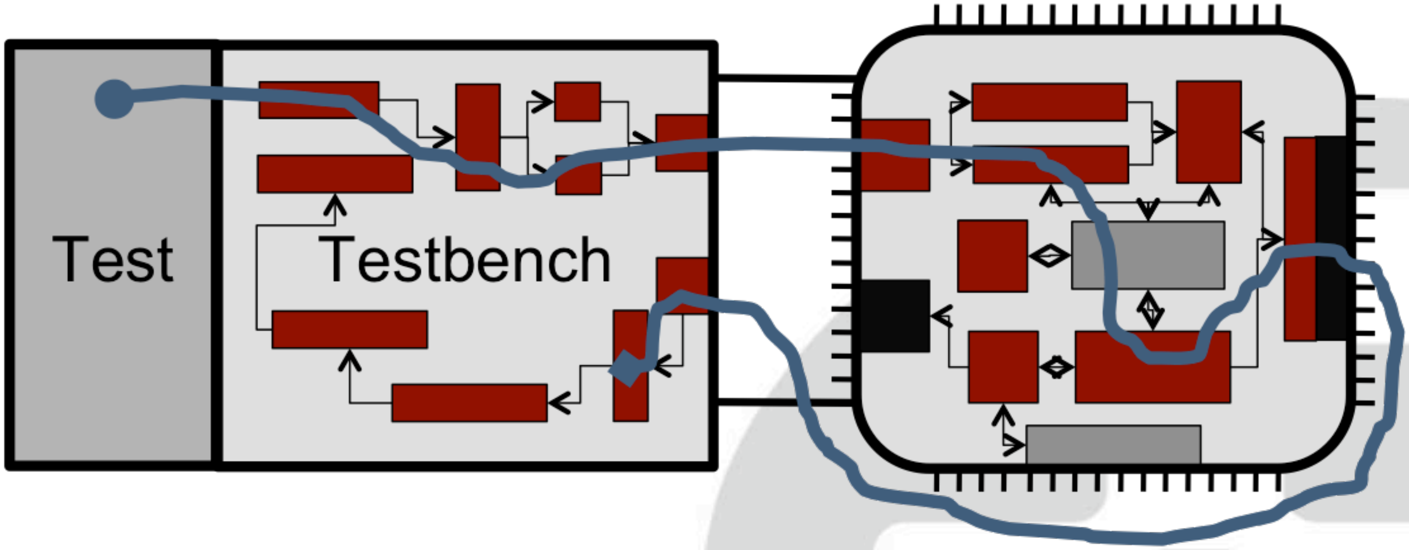
Exercise: Operation Basic Sanity

- Help teams plan a first product increment
  - Start with a finished product



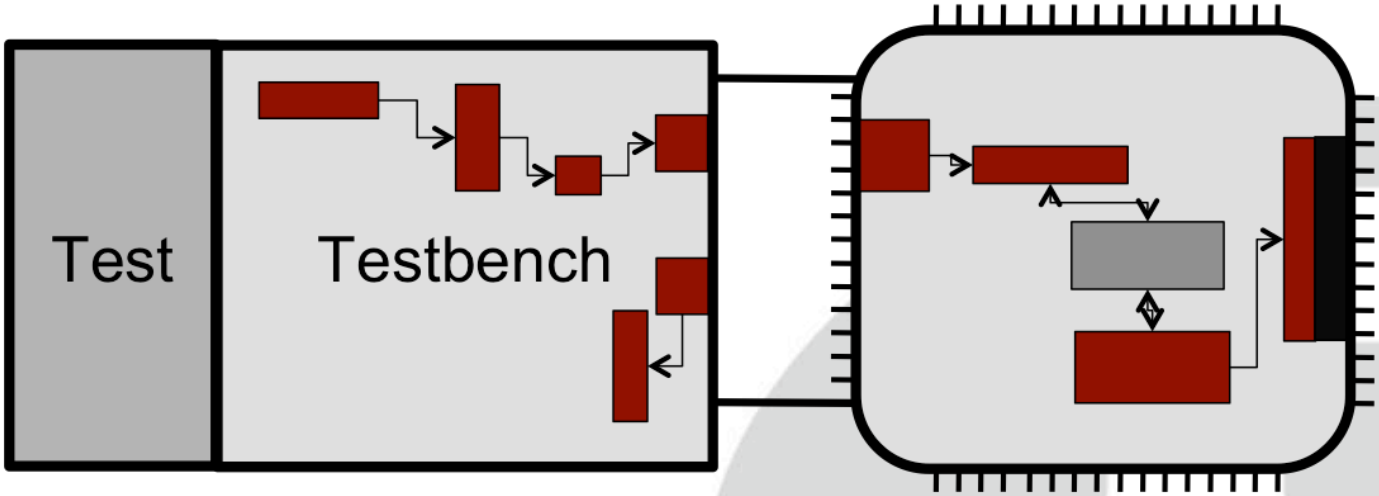
Exercise: Operation Basic Sanity

- Help teams plan a first product increment
  - Start with a finished product
  - Identify the sanity path



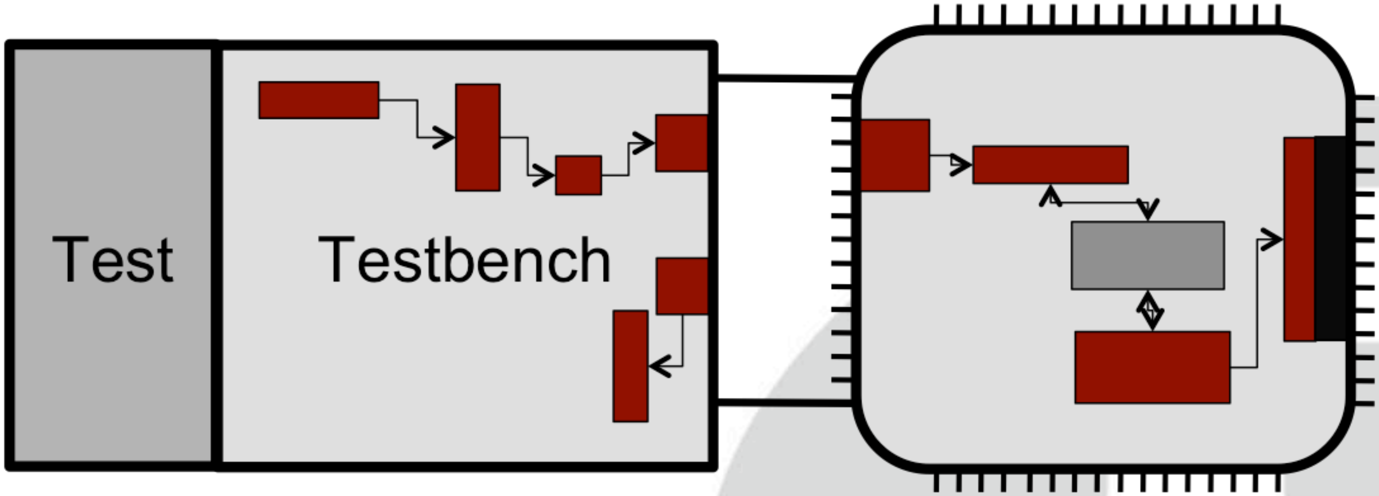
Exercise: Operation Basic Sanity

- Help teams plan a first product increment
  - Start with a finished product
  - Identify the sanity path
  - Remove everything you don’t need



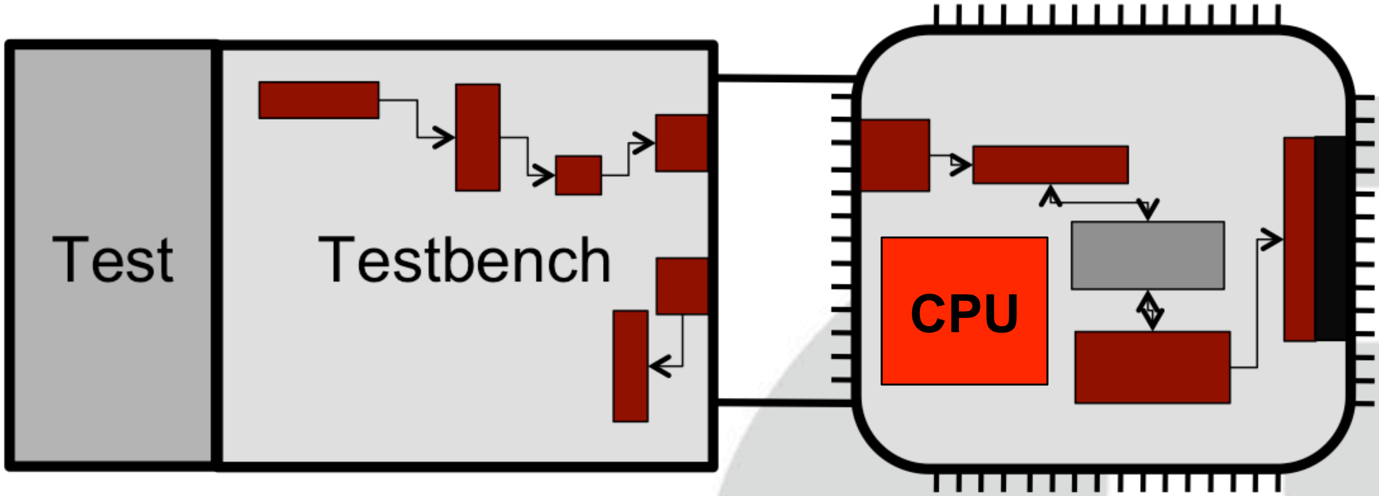
Exercise: Operation Basic Sanity

- Help teams plan a first product increment
  - – Start with a finished product
  - – Identify the sanity path
  - – Remove everything you don’t need
  - – Plan how to build what’s left



Exercise: Operation Basic Sanity

- Help teams plan a first product increment
  - Start with a finished product
  - Identify the sanity path
  - Remove everything you don't need
  - Plan how to build what's left



Summary

- What Are The Strange Hardware People Doing?
  - We're doing big bang development... and it's not working
- Taking the Manifesto Where It Wasn't Meant To Go
  - The manifesto absolutely applies to hardware development
- If it looks like a Duck… And it quacks like a Duck…
  - Present agile in a way that makes sense to hardware developers
- Resources
  - www.AgileSoC.com
    - blog/articles/video
  - open-ended rambling
    - nosnhojn@gmail.com - @nosnhojn
    - bryan.morris.peng@gmail.com - @bryanmorrispeng