Agile FPGA Development

Why FPGA developers should leverage the software ecosystem

David Snowdon, October 2020
daves@arista.com

The Marketing-y Bit

What’s new?

Introducing SwitchApp

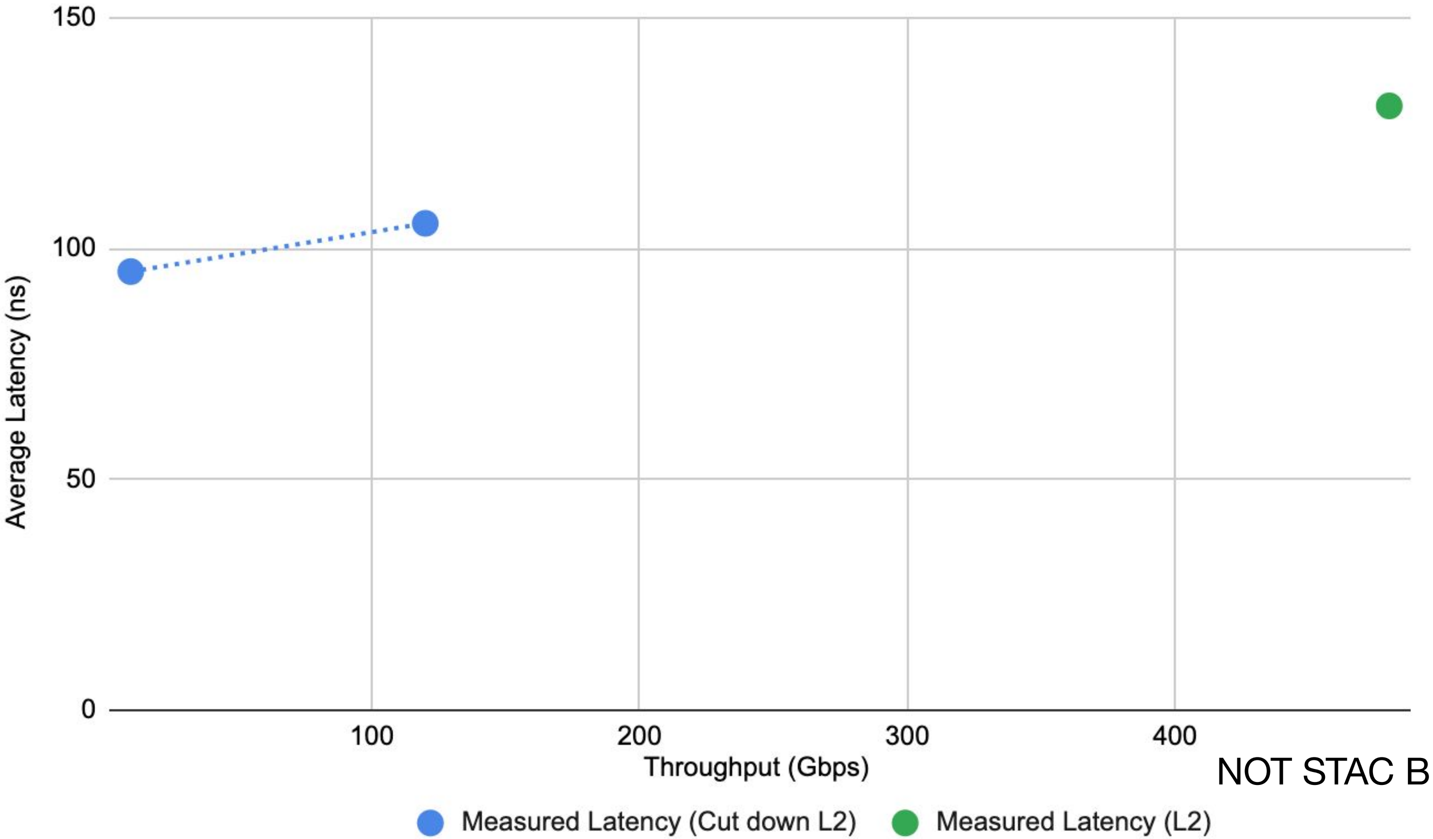
- A full-featured 1/10/40G Layer 2 switching, implemented in FPGA
  - Ultra-low latency packet forwarding in 90-130* ns
  - Non-blocking bandwidth profiles provide up to 480* Gbps
  - 48x 1/10G ports.
  - 10K MAC addresses

- Fully integrated with EOS, running on 7130 LB devices
  - Standard EOS CLI and protocols.

- Available for download and testing on 7130 LB devices today.
  - Talk to us about the details and roadmap.

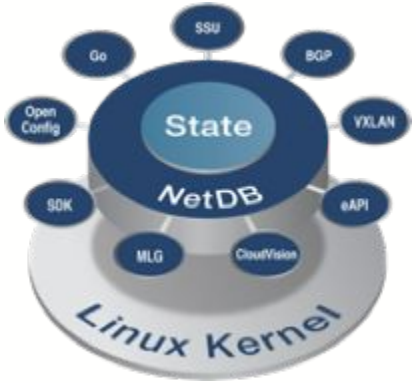
*NOT STAC BENCHMARKS

Introducing SwitchApp



NOT STAC BENCHMARKS

EOS-on-7130 update



- The latest EOS alpha EFT is released!
  - See the download portal.
  - Now supporting most MOS L1 functionality in EOS.
  - GA in Q4

- MetaMux is released as alpha EFT.
  - MetaWatch and MultiAccess close behind.

- Custom application development APIs in EOS in Q4.
  - The FPGA doesn't change.
  - Minimal porting effort.

MetaWatch De-Duplication

- Optionally remove duplicate packets from the MetaWatch captures
  - Substantially reduce the size of a packet stream
  - Various masking options to ignore some headers when detecting duplicates

- Options upon duplicate detection:
  - Flag, drop or truncate (but retain the timestamp and other metadata)

- Use cases:
  - Multicast data capture
  - Capture the same packet at multiple points (e.g. across a switch/router)

FPGA development

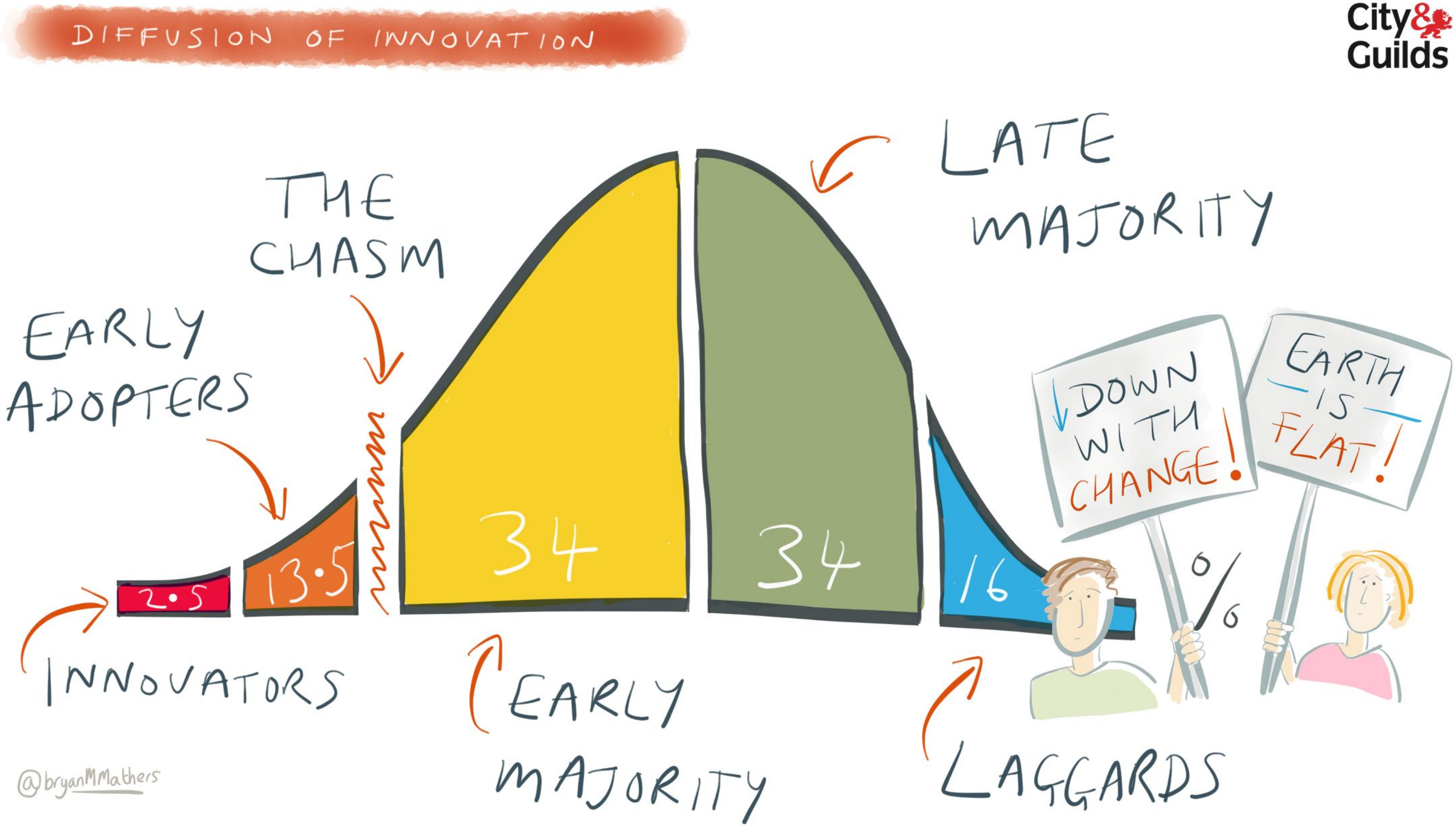
- FPGA developer’s kit
    - Released, and in production with customers
    - Lightweight build system -- just type `make`
    - End-to-end examples
    - Improved Mac/Phy IP core

- Integration with Xilinx’s Vitis toolchain
    - Shell management application
    - Ethernet MAC Kernel
    - Support for Xilinx’s Market Maker Example Design

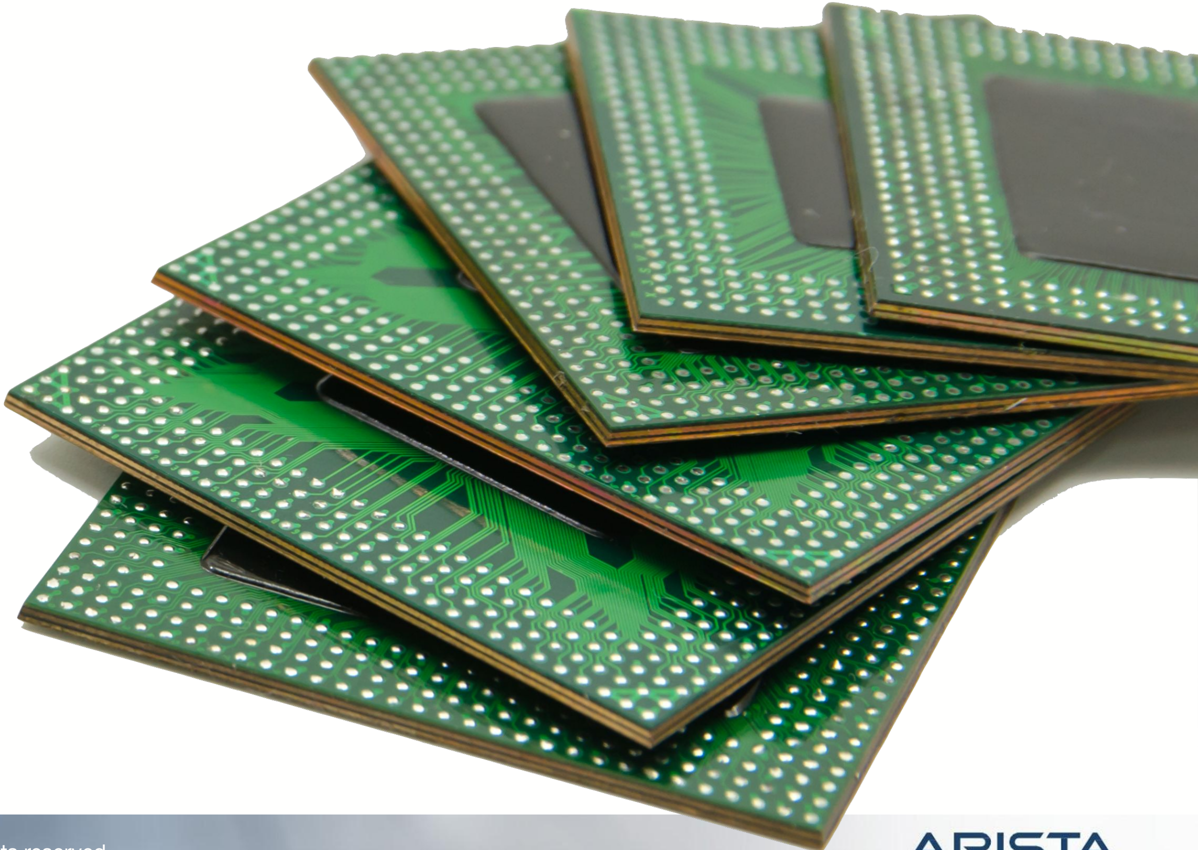
Agile FPGA Development

Why I’m giving this talk

City&
Guilds



Diffusion of Innovation by @bryanMMathers is licenced under CC-BY-ND



Comparing development workflows

CPU

1. Type text into emacs.
2. Run a compiler (gcc?)
3. Get a binary (.o? .bin? .rpm?)
4. Run the binary.

FPGA

1. Type text into emacs.
2. Run a compiler (Vivado? Quartus?)
3. Get a binary (.bit?, .pof? .rpm?)
4. Run the binary.

FPGA developers are software developers

Comparing terminology

CPU	FPGA
Software Engineer	Hardware Engineer
Unit test	Testbench, simulation
Testing	Verification
Library	IP Core

Comparing programming languages

CPU	FPGA
Python	
C++	OpenCL
C	SystemVerilog
Assembly	RTL -- Verilog, VHDL
Binary	FPGA Editor, Netlist

People who write code for FPGAs are software engineers.

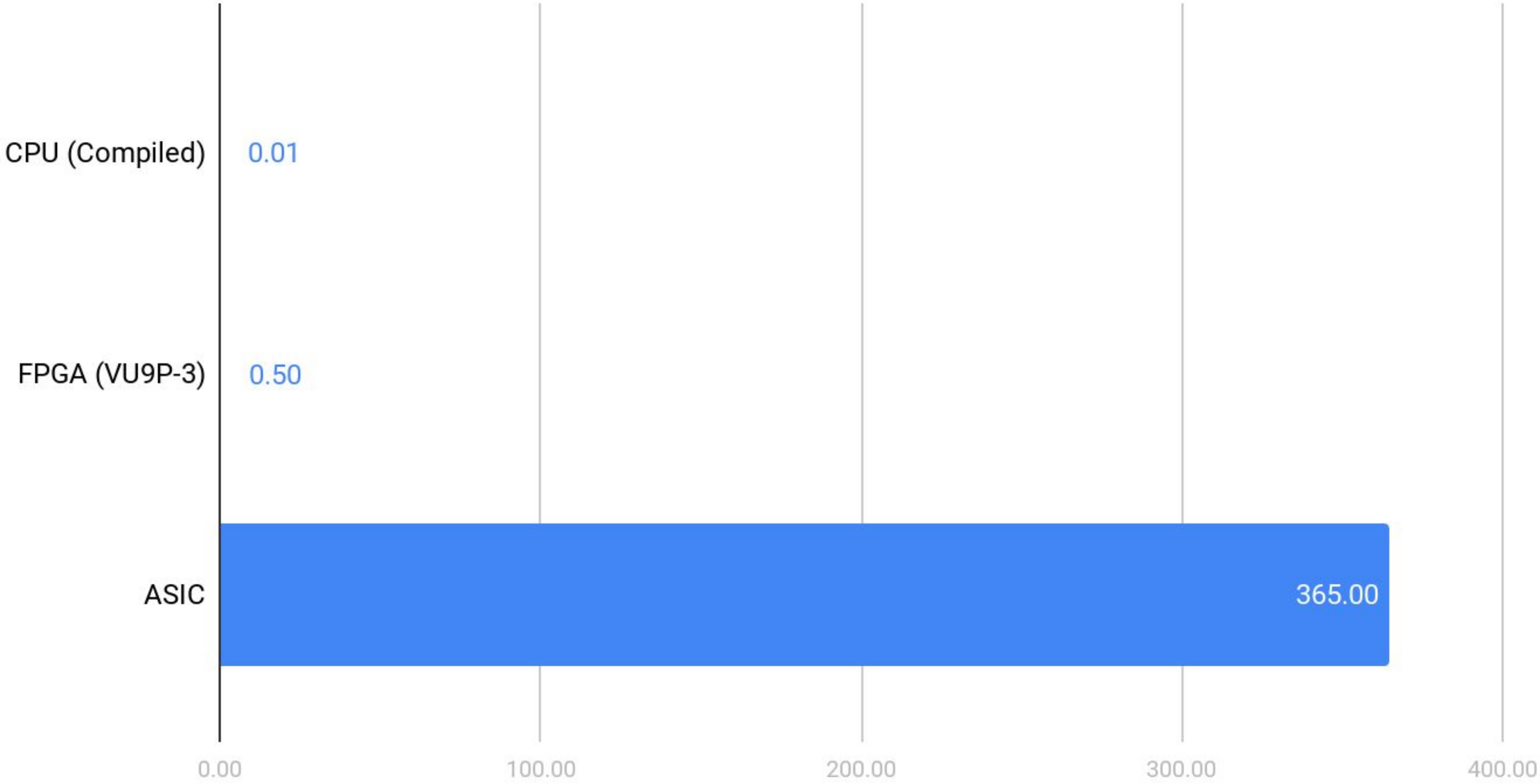
This is what ***real*** hardware looks like...

A few real differences...

- The virtual machine is pretty different.
  - The tooling to make FPGA programs easier to write and comprehend is getting better.

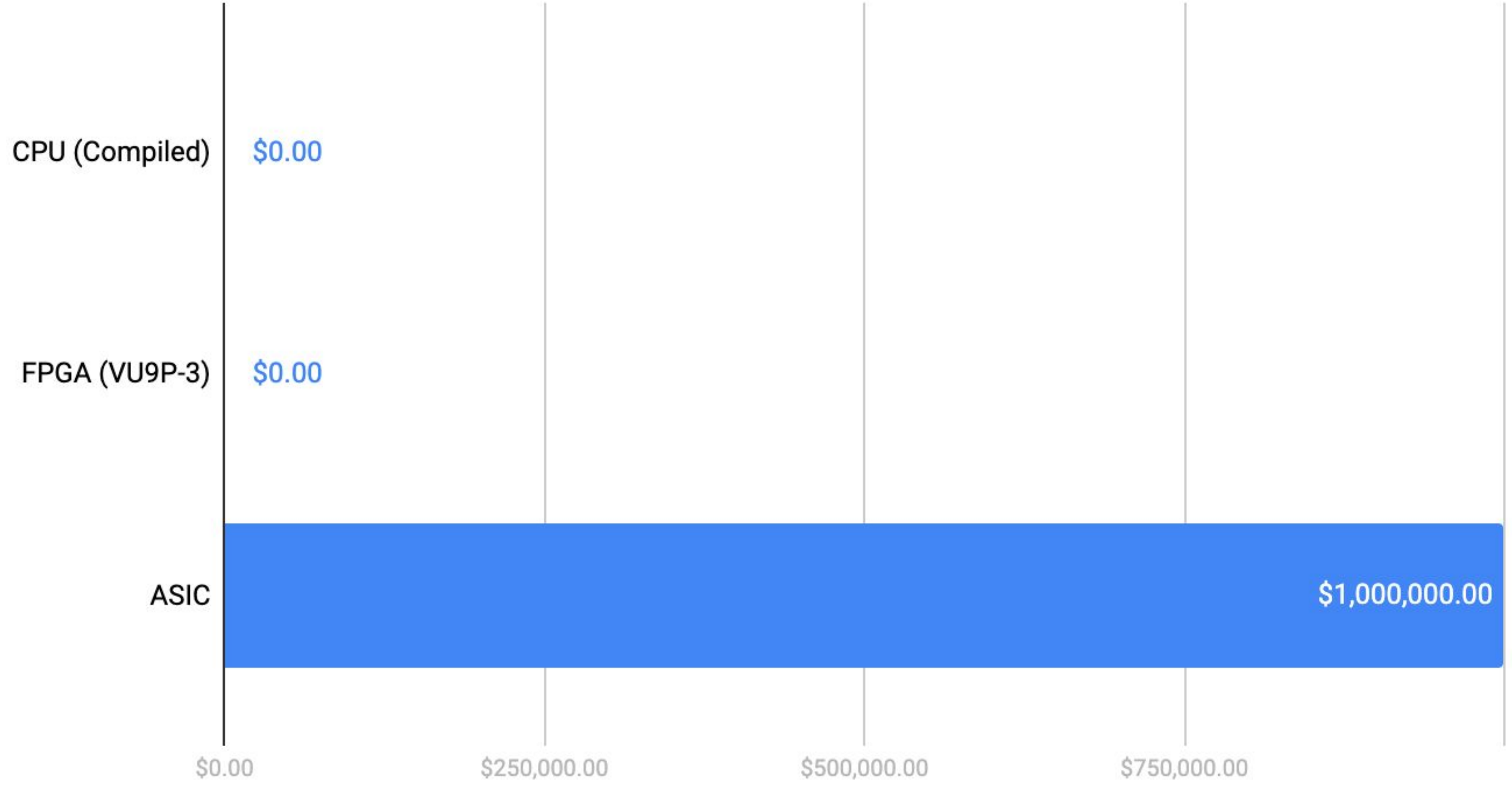
- Build times are long…
  - But *way* shorter than ***real*** hardware.

Hypothetical Build times (Days)



NOT STAC BENCHMARK

Hypothetical Incremental Build cost (Dollars)

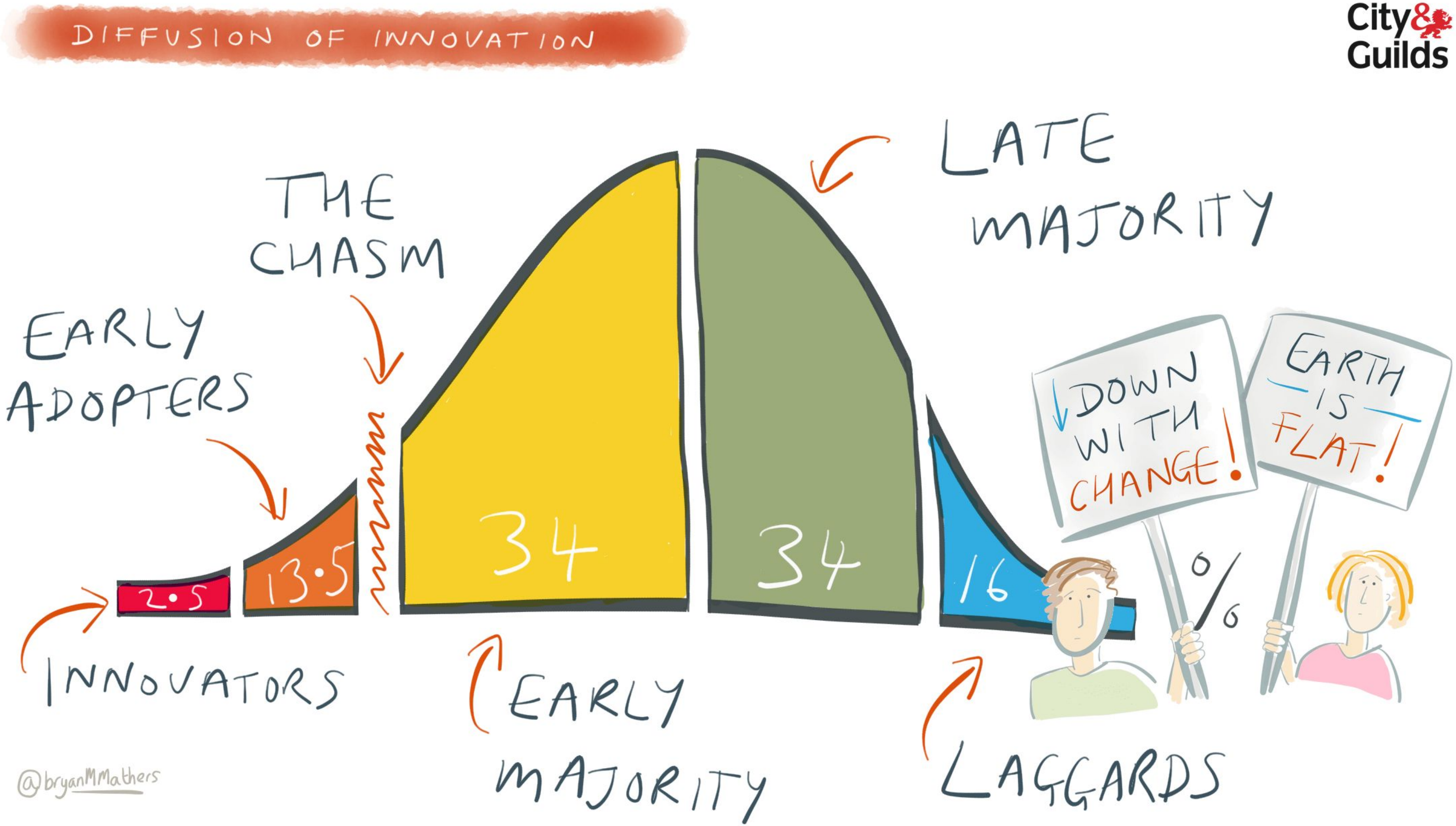


ARK

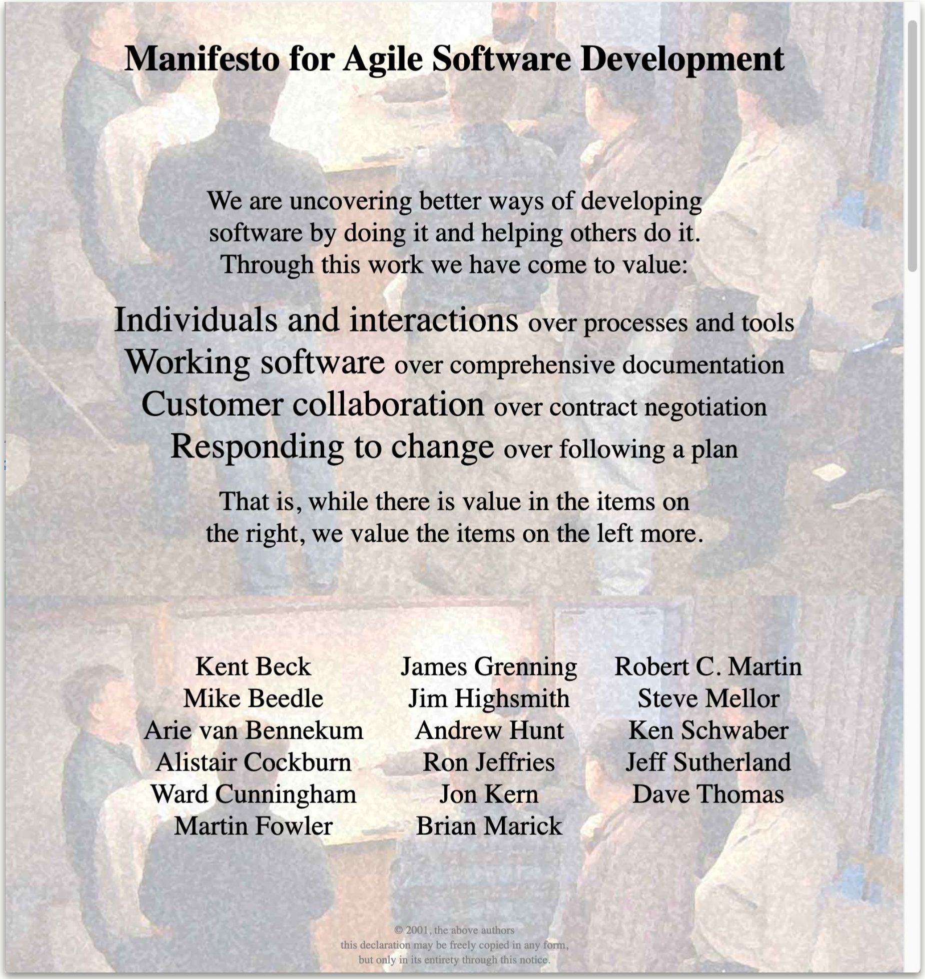
A few real differences...

- The virtual machine is pretty different.
  - But then… So is the VM for a GPU.
- Build times are long…
  - But *way* shorter than ***real*** hardware.
- FPGA code has to meet “timing constraints”.
  - Signals take time to propagate in an FPGA -- if the tools can’t get a particular piece of code to work, they’ll throw an error
  - Then again, this is true of register allocation in a CPU (the compiler hides a lot).
- Much less infrastructure, less mature.

City&Guilds



Diffusion of Innovation by @bryanMMathers is licenced under CC-BY-ND



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.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Agile FPGA development

- ASICs are designed using a waterfall methodology
  - rigid specifications and designs, separated build/test teams
  - Everything that Agile is not.

- FPGA developers are often ex-ASIC developers
  - Design from the ground up, long development cycles.

- Modern software is written using agile techniques
  - Rapid iteration means we can respond quickly (e.g. getting our algorithms to production more quickly)

⇒ Treating FPGAs as software-programmable brings FPGAs into the software ecosystem.

Agile Principles

https://agilemanifesto.org/

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

How can FPGA development be more agile?

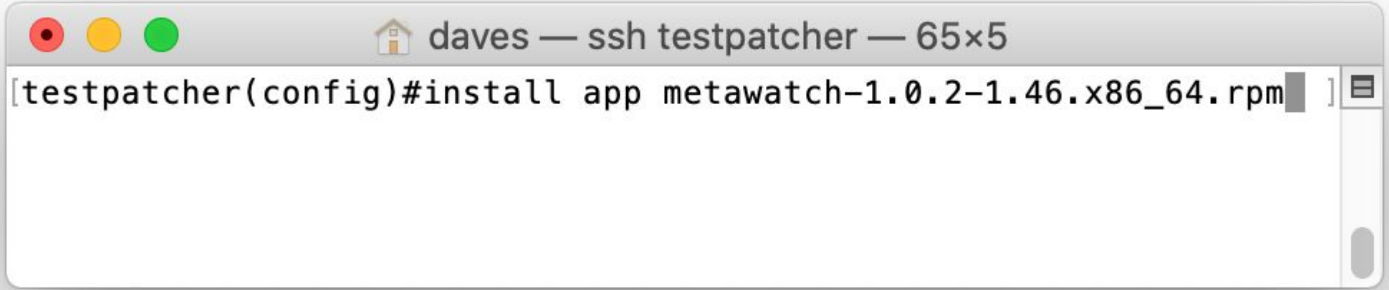
- Re-use
- Improved system architectures
- Faster Deployment
- Build/development Tooling:

How can FPGA development be more agile?

- Re-use
    - Operating System Software
    - Libraries/IP Cores
    - Reusable test infrastructure
    - Open-source software

How can FPGA development be more agile?

- Faster Deployment
  - Packaging and package management matters -- RPM, .swix
  - Deployment automation
  - Containerisation
  - Continuous deployment
  - Clouds -- FPGA-as-a-service
  - Orchestration -- Kubernetes



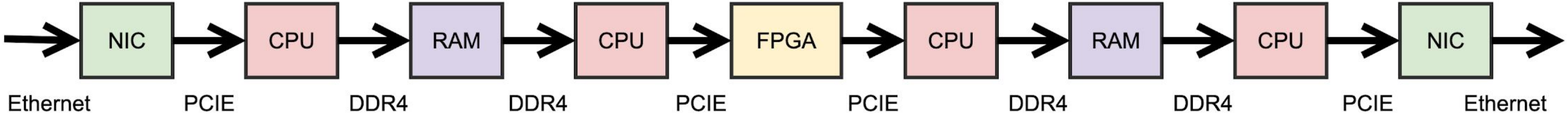
```
testpatcher(config)#install app metawatch-1.0.2-1.46.x86_64.rpm
```

How can FPGA development be more agile?

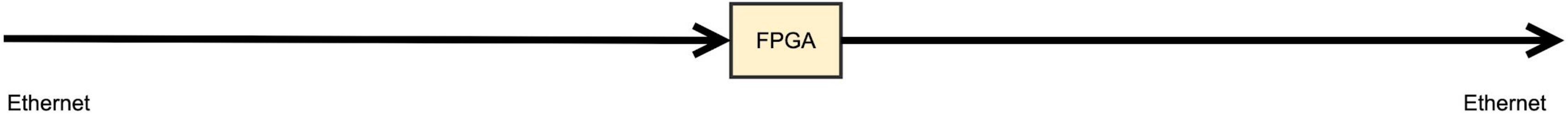
- Modern System Architectures
  - Every system is a distributed system
  - Microservices
  - Function as a service

A little personal observation...

- Most FPGAs are subservient to a CPU
  - Clusters of FPGAs are formed by pairing many FPGAs with a CPU.
  - This is crazy!



- HFTs build network-attached FPGA based solutions
  - We are nearly always a bump-in-the-wire.
  - This applies in all kinds of other use cases…



A little personal opinion...

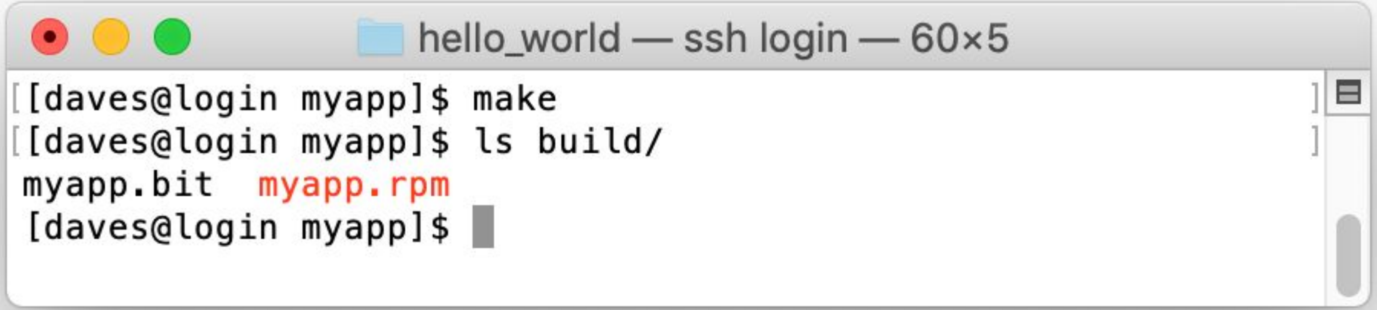
- PCIE is the worst… *(for attaching FPGAs)*
  - Hard to re-program;
  - Hard to debug;
  - Hard to secure;
  - Ability to hard lock the host CPU;
  - Fixed relationship between the CPU and the FPGA (not very agile).

⇒ *Ethernet is awesome.*

- Communicate with the FPGA(s) as a microservice.
- Debug using network monitoring techniques.

How can FPGA development be more agile?

- Build/development Tooling:
  - Repeatable builds!
  - Version control
  - Linting
  - Code generation
  - Automated build and test systems
  - Continuous Integration
  - Containerised builds



```
[daves@login myapp]$ make
[daves@login myapp]$ ls build/
myapp.bit  myapp.rpm
[daves@login myapp]$ 
```

Concluding recommendations...

- Ditch PCIE -- Talk to FPGAs and CPUs alike, via the network.
- Stop diminishing an FPGA as an “accelerator”.
- Embrace the software paradigm -- FPGAs are software programmable resources, so we can apply the agile principles to great effect.
- Embrace the software ecosystem -- Cross the chasm by using the tools which power software engineering.

Last thing…

- We (the STAC community) have the ability to lead the world in this area.

Thank You

www.arista.com