Containerize Your Chip Development Environment Using Docker

Chris Drake
Google

March 30, 2016
Santa Clara, CA

About Me

- Intel 2005-2014
- Currently at Google

Agenda

Introduction
Linux Containers
Docker Fundamentals
“Devshell” Methodology
Performance Data
Final Thoughts
Questions

Google

snug
Synopsys Users Group
Silicon Valley

Introduction

System-on-Chip Development

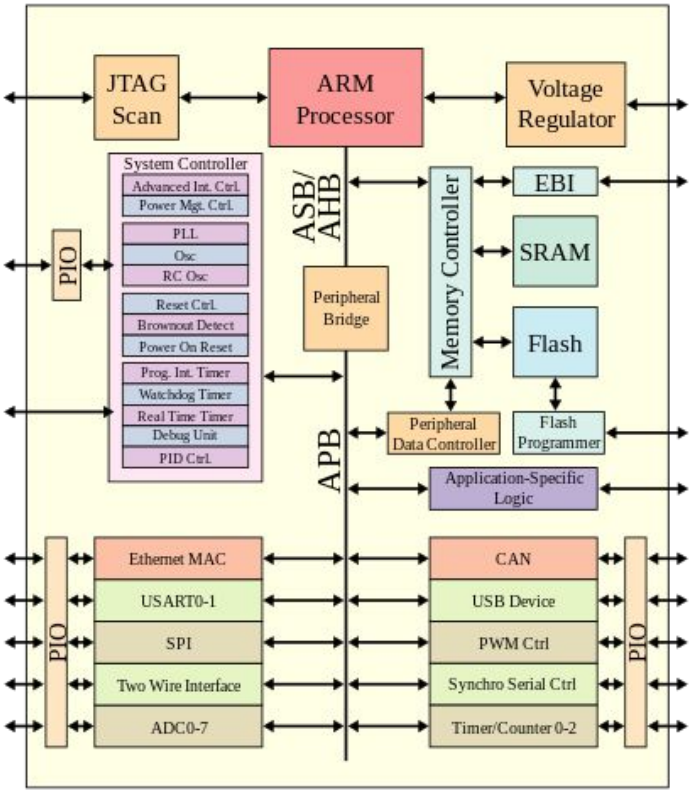
In-House design/verif IP

Third Party design/verif IP

Design for Reuse

Dependencies:

- System (RedHat, SLES, ...)
- System tools (gcc, perl, …)
- System libs (glibc, libfftw3, …)
- EDA tools (VCS, DC, …)
- EDA libs (UVM 1.0, 1.1, 1.2)
- Applications / Simulators



Pain Points

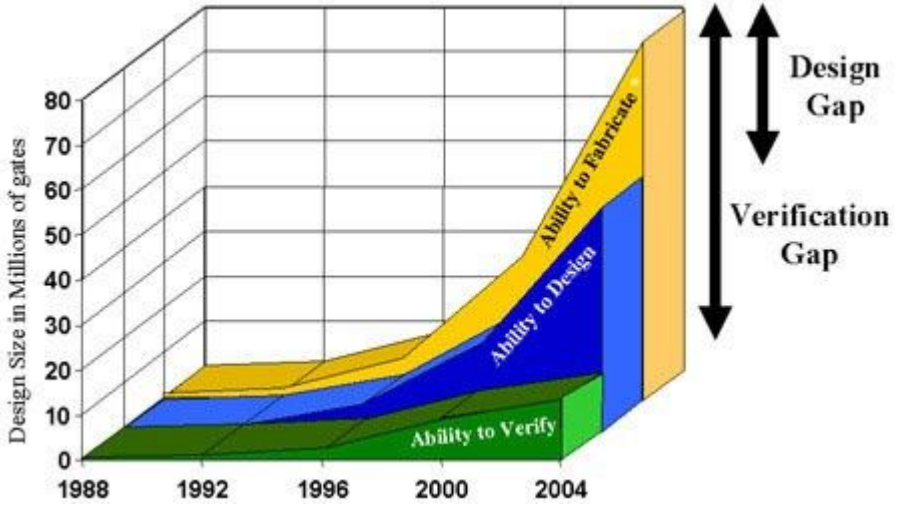
System administrator / CAD engineer “matrix of hell”:

- Hardware generation
- OS distribution
- System packages
- Application dependencies
- Applications (new version of EDA tool X every few weeks)

Pain Points, Cont'd

Verification Crisis:
Our ability to verify has not kept up with our ability to design and fabricate.

Indicator:
http://danluu.com/cpu-bugs/



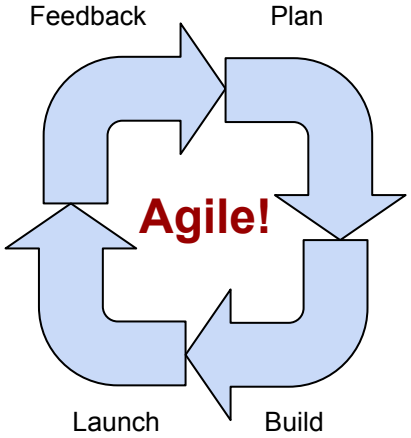
Source: EETimes

Agile Hardware Development?

(as opposed to waterfall)

1. Early and continuous delivery
2. Welcome changing requirements
3. Deliver frequently

4-12…



The *development environment* is as important a part of the continuous integration process as *project source code*.

See: David Patterson's EETimes Article, [Agile Design for Hardware]

Motivating Questions

1. How much time spent debugging arcane dependency errors, eg: “`/usr/lib64/libstdc++.so.6: version `GLIBCXX_3.4.9' not found`”

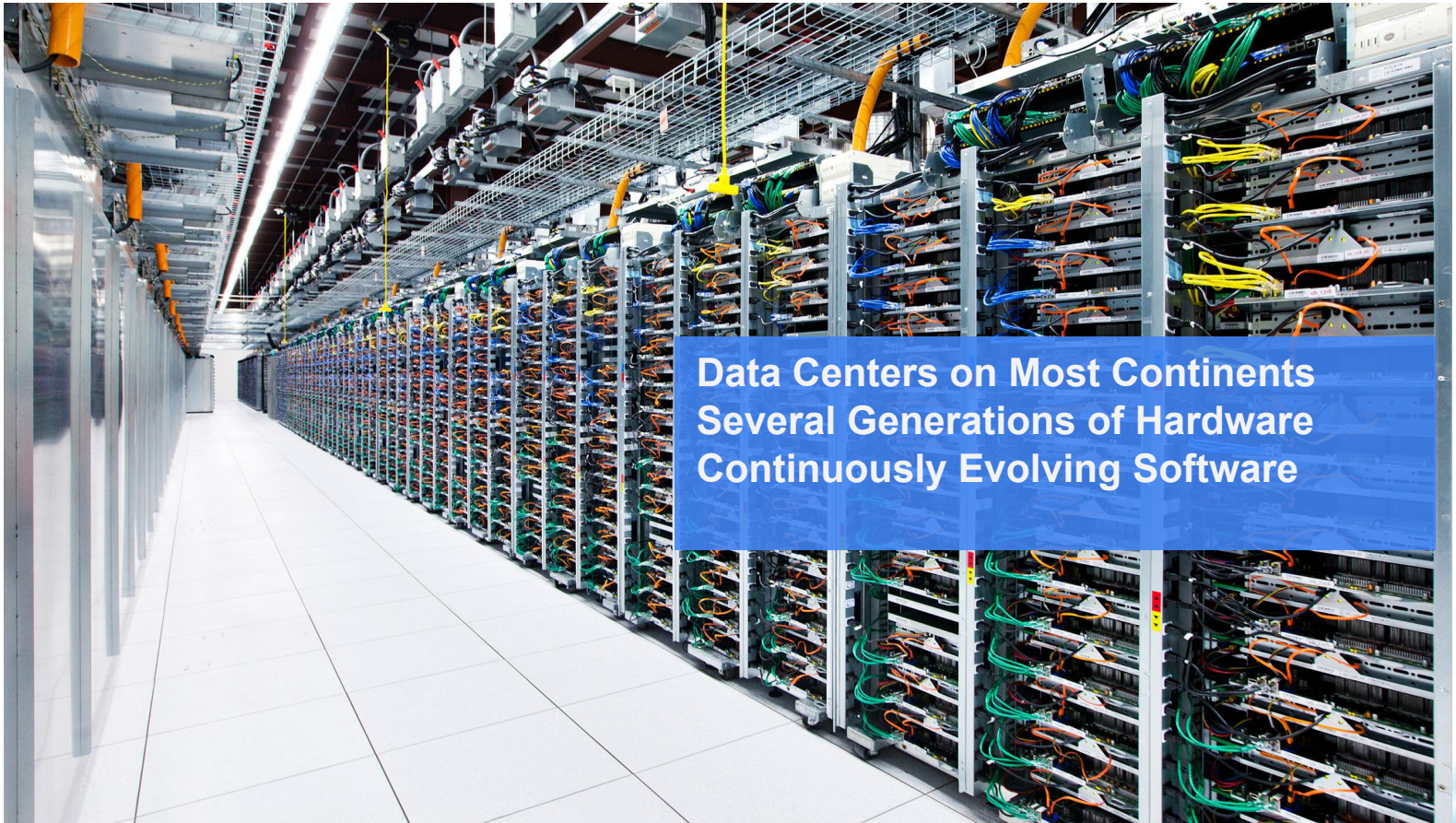
2. When collaborating with third parties, how much effort required to reproduce the build environment?

3. After taping out chip A, then reopening work on chip A’, do all the build/test scripts still work now that the systems are different?

4. For AEs, how much effort is required to get clean test cases from customers? What dependencies are required to make it work?

5. Do you maintain an old-growth NFS forest of tools and libraries?

Parallels From Google's Perspective



Question

How do I package my development environment so that it can be moved around from machine to machine?

Containers!

Linux Containers

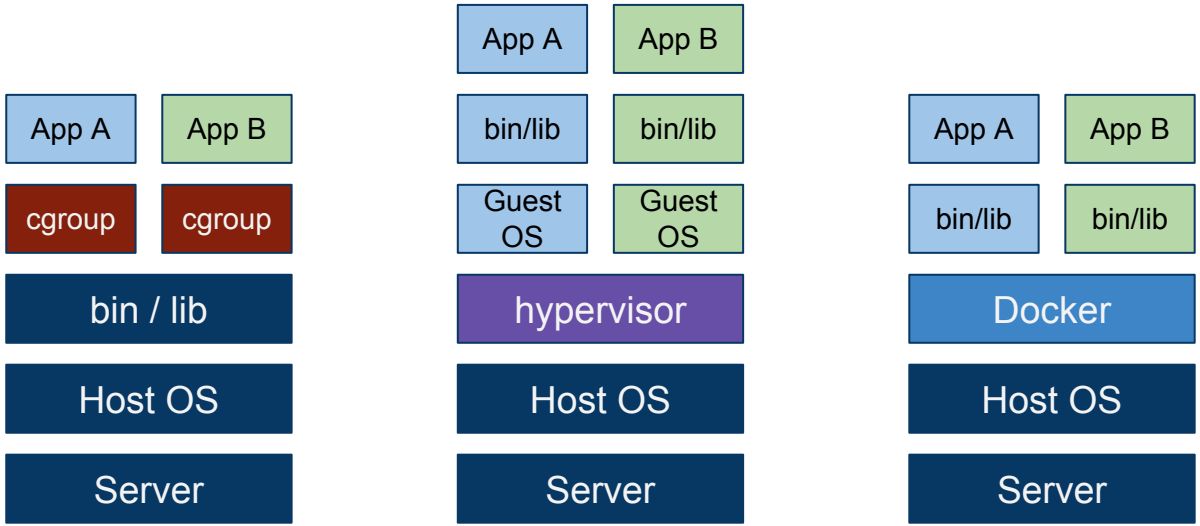
Combination of two things:

1. Runtime resource (cpu/mem/etc) isolation
2. An image - the files that make up the applicaton

See https://linuxcontainers.org/ for more info about LXC

See Borg, Omega, and Kubernetes - Lessons Learned from Three Container Management Systems Over a Decade, by B. Burns et al.

Comparison of Virtualization Solutions



OS Control Group **Virtual Machine** **Docker**

Everything at Google runs in containers:

- Gmail, Web Search, Maps, ...
- MapReduce, batch, ...
- GFS, Colossus, ...
- Even **Google's Cloud Platform:** our VMs run in containers!

We launch over **2 billion** containers **per week**

Shipping Containers At Clyde, by Steve Gibson

Google's Solution

Borg - cluster manager

Linux, cgroups-based resources container (like lmctfy)

Midas Package Manager (video)

Ref: Borg Paper

"Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines."

Google

snug
Synopsys Users Group
Silicon Valley

Docker Fundamentals

Containers in the Wild: Docker

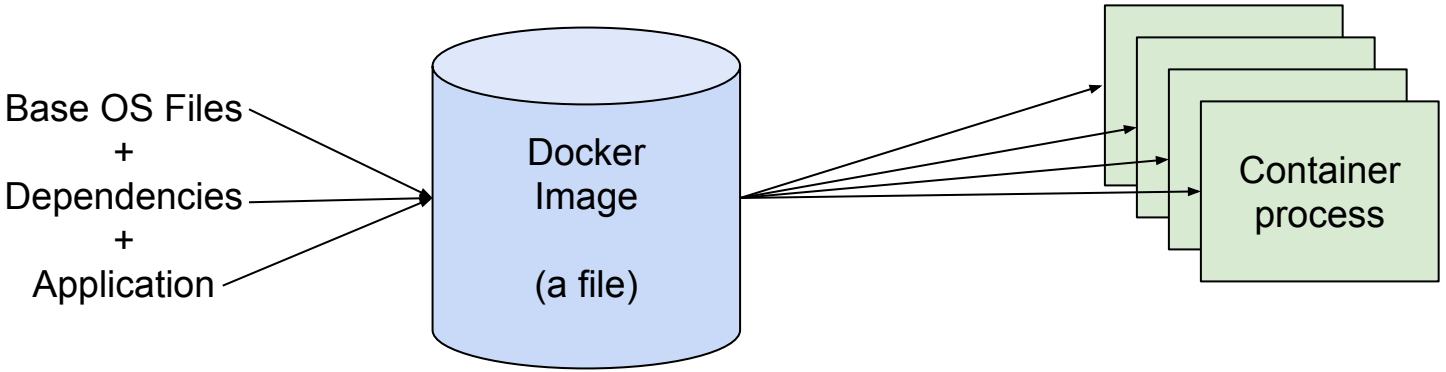
“Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.”

“Build once, run anywhere.”

Docker Work Flow

1. Docker **build** step

2. Docker **run** step



Create an Image

```
1 FROM centos:6
2 RUN yum install -y bc gcc-c++ perl tar tcsh
3 RUN yum install -y compat-libtiff3.i686 compat-libtiff3.x86_64 \
4         fontconfig.i686 fontconfig.x86_64 glibc-devel.i686 glibc-devel.x86_64 \
5         libjpeg-turbo.i686 libjpeg-turbo.x86_64 libmng.i686 libmng.x86_64 \
6         libpng12.i686 libpng12.x86_64 libstdc++-devel.i686 libstdc++-devel.x86_64 \
7         libSM.i686 libSM.x86_64 libX11-devel.i686 libX11-devel.x86_64 \
8         libXScrnSaver.i686 libXScrnSaver.x86_64 libXext.i686 libXext.x86_64 \
9         libXft.i686 libXft.x86_64 libXi.i686 libXi.x86_64 \
10        libXrandr.i686 libXrandr.x86_64 libXrender.i686 libXrender.x86_64 \
11        ncurses-devel.i686 ncurses-devel.x86_64
12
13 COPY vcs-mx_vK-2015.09-SP1_common.spf vcs-mx_vK-2015.09-SP1_amd64.spf \
14      vcs-mx_vK-2015.09-SP1_linux.spf vcs-mx_vK-2015.09-SP1_SI32.tar \
15      SynopsysInstaller_v3.2.run /root/
16
17 ENV SYNOPSYS /usr/local/cad/synopsys
18 ENV VCS_VERSION K-2015.09-SP1
19 ENV VCS_HOME $SYNOPSYS/vcs-mx/$VCS_VERSION
20 RUN ./SynopsysInstaller_v3.2.run -dir .
21 RUN yes | ./installer -install_as_root -batch_installer -source . -target $SYNOPSYS
23 ENV PATH $VCS_HOME/bin:$PATH
```

Figure 2: Example Dockerfile to Install VCS-MX K-2015.09-SP1

Save and Load Images

From / To a Repository (dockerhub, gcr.io, quay.io, etc):

- `docker pull REPO/IMAGE:TAG`
- `docker push REPO/IMAGE:TAG`

From / To an archive file:

- `docker load -i /path/to/image.tar`
- `docker save -o /path/to/image.tar`

Start the Container

```
docker run [options] IMAGE [command] [arg ...]
```

Lots of options, eg:

- `-i, --interactive`
- `-t, --tty`
- `-v, --volume`
- `--cpu-period, --cpu-quota`
- `-m, --memory`
- `--net`
- `-p, --publish`
- `-e, --env`

Container Ecosystem

Google

snug
Synopsys Users Group
Silicon Valley

Google Container Engine

Amazon EC2

Docker Data Center

Kubernetes

Jenkins

Mesos

LSF / GridEngine / SLURM / Torque / etc

… and many others

Google

snug
Synopsys Users Group
Silicon Valley

Devshell Methodology

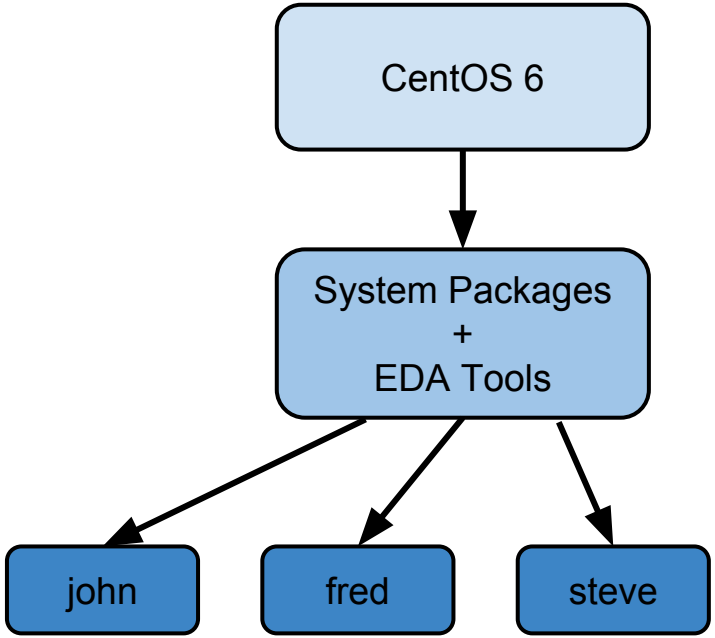
Challenges with Default Containers

Shell is started as user “root”

No external files accessible (ie, it’s a jail)

X11 forwarding is not enabled

Development Shell Methodology



Base Layer from DockerHub

Add application files

Use devshell to
customize per user

Mount external volumes
Mount network, X11

Trim IT-induced Fat

Google

snug
Synopsys Users Group
Silicon Valley

Imagine you have received 3rd party IP w/ C++ GRM, Perl/Python scripts, etc

Previously:

- Install application for a particular OS
- Make sure you have the right version of this tool, that library, etc
- Make sure your Perl/Make/GCC, and $LD_LIBRARY_PATH are correct'ish
- Compile. Pray.
- Explain that procedure to all developers on the team

With Docker:

- Give developer any Linux machine w/ Docker
- Execute the devshell script: identical dev environments

Google

snug
Synopsys Users Group
Silicon Valley

Performance Data

Does It Degrade Performance?

Performance Considerations: Build

Image	Build Time (s)
Base CentOS-6 plus system dependencies	320
Devshell layer	20

Table 1: Docker Image Build Times

These are *one-time* costs,
and will vary wildly between systems.

Container “Boot” Time

Virtual machines take ~minutes to boot.

Docker container start time is negligible (~ms).

Performance Considerations: VCS-MX

Parameters	Host Runtime (s)	Docker Runtime (s)
NUM_BLOCKS=10 NUM_VECS=12,000	56.65	56.59
NUM_BLOCKS=10 NUM_VECS=120,000	548.88	548.67

Table2: Simulation Performance Comparison

Google

snug
Synopsys Users Group
Silicon Valley

Final Thoughts

Next Steps

- This talk focused on the interactive debug use case
- Kubernetes - run containers at scale!
- First steps towards cloud HPC
- Don’t like Docker?, try Rocket

Plenty of players in this space:

- Mesosphere
- IBM LSF
- Univa GridEngine
- AdaptiveComputing Torque
- SchedMD SLURM
- Cloud Native Computing Foundation (https://cncf.io/)

Caveats

User requires root access to a Linux development machine
(or a service that can act as root on the user's behalf)

Docker minor releases sometimes break backwards compatibility

Containerizing at scale comes at a cost

Try It Out for Yourself

Notes:

- GitHub Repo with example RTL and devshell script: https://github.com/cjdrake/AES

Readdressing Questions

1. How much time spent debugging arcane dependency errors, eg: “`/usr/lib64/libstdc++.so.6: version `GLIBCXX_3.4.9' not found`”

2. When collaborating with third parties, how much effort required to reproduce the build environment?

3. After taping out chip A, then reopening work on chip A’, do all the build/test scripts still work now that the systems are different?

4. For AEs, how much effort is required to get clean test cases from customers? What dependencies are required to make it work?

5. Do you maintain an old-growth NFS forest of tools and libraries?

Readdressing Questions

When Your Development Environment is a Docker image:

1. How much time spent debugging arcane dependency errors, eg: "`/usr/lib64/libstdc++.so.6: version `GLIBCXX_3.4.9' not found`"

   None

2. When collaborating with third parties, how much effort required to reproduce the build environment?

   None

3. After taping out chip A, then reopening work on chip A', do all the build/test scripts still work now that the systems are different?

   Trivial

4. For AEs, how much effort is required to get clean test cases from customers? What dependencies are required to make it work?

   Easy

5. Do you maintain an old-growth NFS forest of tools and libraries?

   Allow some pruning

Summary

- Web developers have popularized containers to separate concerns between platform and application
- A popular, open source container implementation, Docker, can be adapted for EDA
- SoC design teams can streamline their devops with Docker

Google

snug
Synopsys Users Group
Silicon Valley

Questions?

Google

snug
Synopsys Users Group
Silicon Valley

Thank You

Google

snug
Synopsys Users Group
Silicon Valley

Backup