FireSim

A Brief Tour of FireSim: The Manager & Compiler; Building Hardware Designs

https://fires.im

@firesimproject

ASPLOS 2022 Tutorial

Speaker: Sagar Karandikar

Berkeley Architecture Research

Agenda: What Will We Cover?

1) The Compiler → Golden Gate

- Invoke it on example RTL
- Inspect its outputs

2) The Manager → `firesim`

- Explain how it's configured
- Demonstrate how it's used to build bitstreams

Where is FireSim in Chipyard?

With the software RTL simulators!

~/chipyard-afternoon/sims/firesim

→ This has been exported as $FDIR

Interactive:

```
# <ssh back onto your ec2 instance>

$ tmux new -s afternoon

$ cd $FDIR

$ ls
```

FireSim's Directory Structure

`sim/`

- Golden Gate lives here
- Scala & C++ sources for additional FireSim models
- Make-based build system to invoke Golden Gate

`deploy/`

- Manager lives here
- FireSim workload definitions

`platforms/` → AWS FPGA/Vivado project definitions

`sw/` → target software & FireMarshal (more on this later)

Agenda: What Will We Cover?

1) The Compiler → “Golden Gate”

- Invoke it on example RTL
- Inspect its outputs

2) The Manager → `firesim`

- Explain how it’s configured
- Demonstrate how it’s used to build bitstreams

Interactive:

```
$ cd $FDIR/sim

$ make DESIGN=FireSim
```

An Analogy

- Golden Gate is like Verilator but for FPGA-accelerated simulation

Verilator generates C++ sources to simulate your design.

→ Compile and run on a CPU-host

Golden Gate generates C++ & Verilog to simulate your design.

→ Compile and run on a hybrid CPU & FPGA host

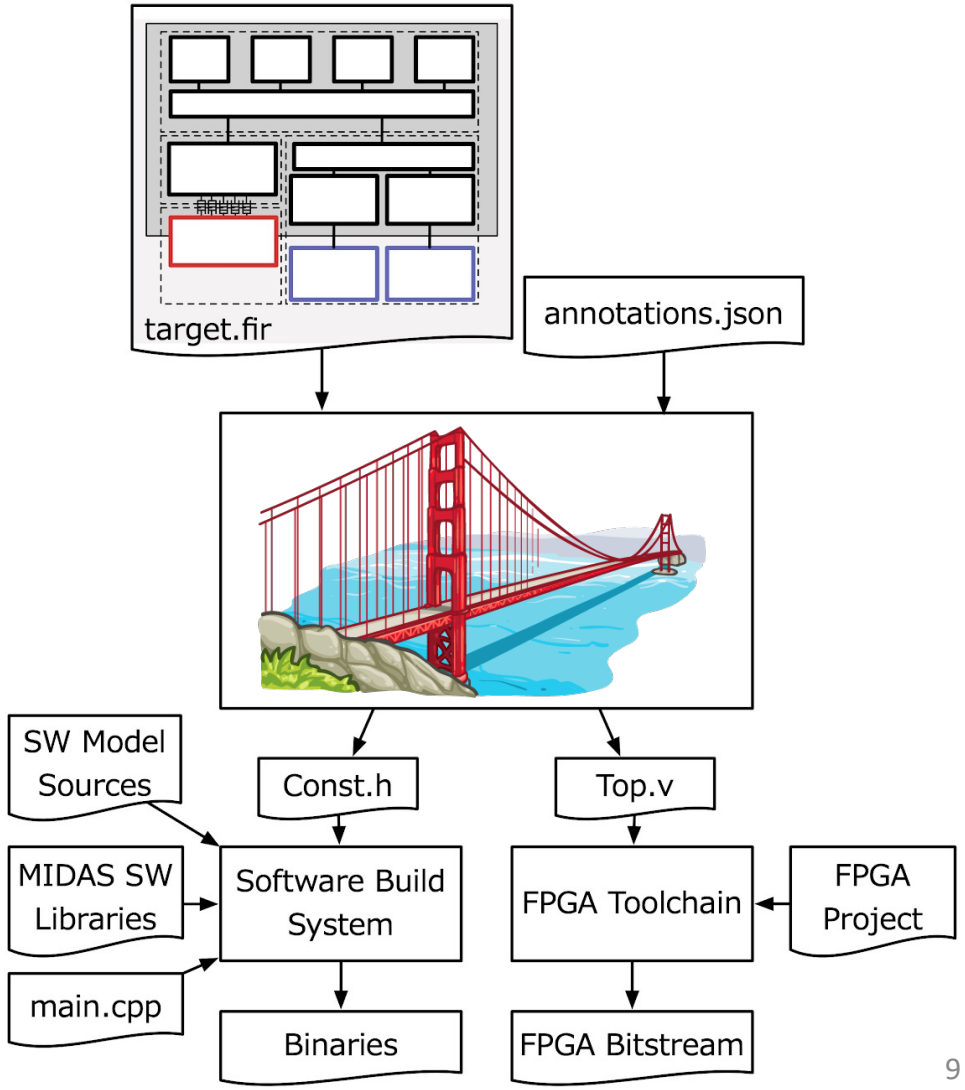
Golden Gate Compiler

Inputs:

- FIRRTL & annos from a Chipyard generator
- Compiler configuration

→ Produces sources for a simulator that are:

- deterministic
- support co-simulation of software models
- *area-optimized to fit more on the FPGA*



Interacting with Golden Gate via Make

- Make invokes Golden Gate with three variables (the “Tuple”):

DESIGN :

- The top level module → MODEL in Chipyard

TARGET_CONFIG:

- The generator’s config → CONFIG in Chipyard

PLATFORM_CONFIG:

- Compiler options passed to Golden Gate

Interactive:

```
$ cd $FDIR/sim/generated-src/f1

# here you'll find output directories for all builds

$ cd <any-directory-here>

$ ls
```

Inspecting the Outputs

`<long-name>.fir & <long-name>.anno.json`

- Target's FIRRTL & annotations

`FireSim-generated.sv`

- The compiled simulator

`FireSim-generated.const.h`

- Simulator's memory map

`FireSim-generated.runtime.conf`

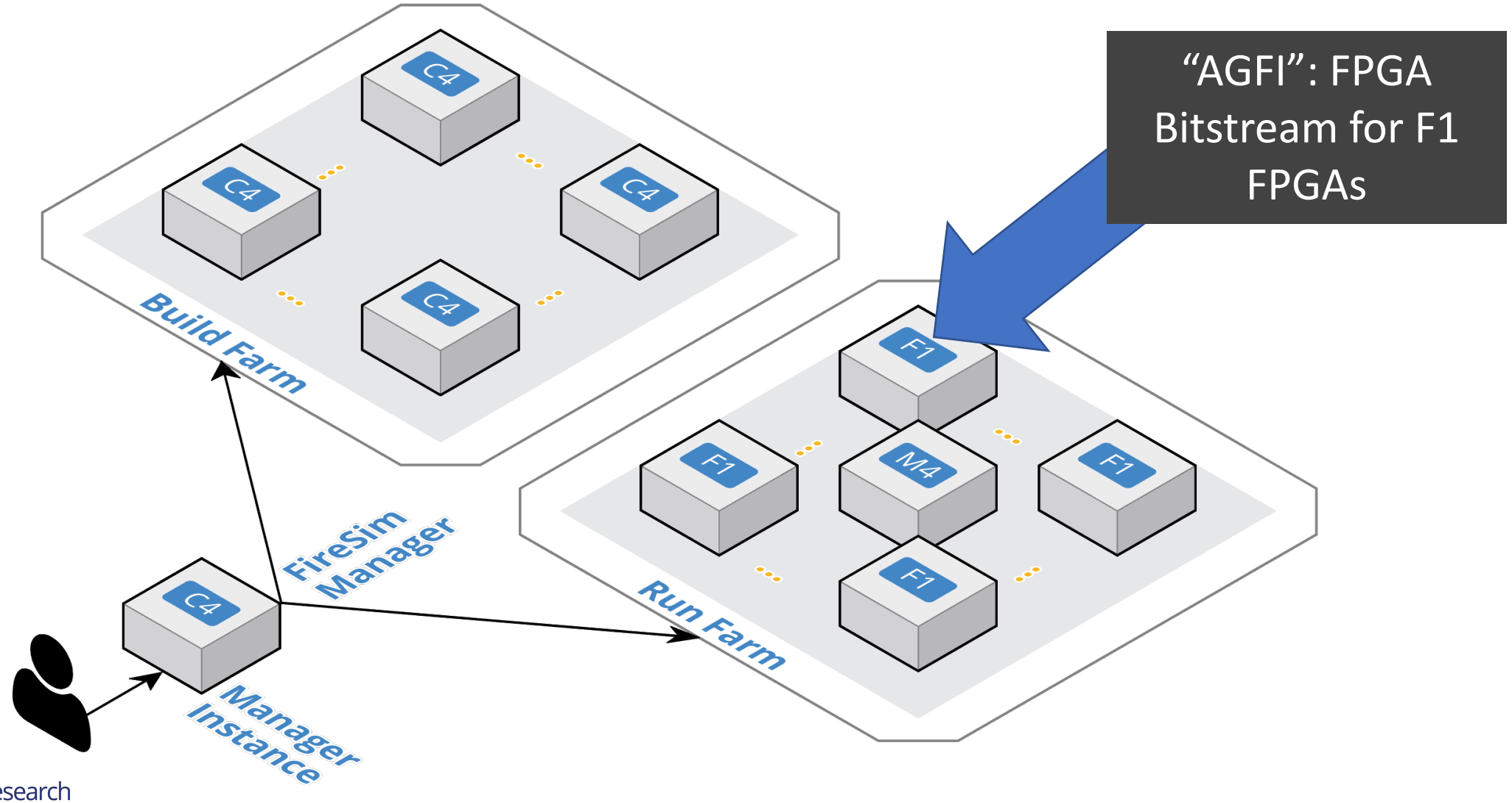
- A default runtime configuration for simulation

Agenda: What Will We Cover?

1) The Compiler → Golden Gate
- Invoke it on example RTL
- Inspect its outputs

2) The Manager → `firesim`
- Explain how it's configured
- Demonstrate how it's used to build bitstreams

Background Terminology



Using the `firesim` Manager Command Line

- sourcing `sourceme-f1-manager.sh` puts `firesim` on your path
- can call `firesim` from anywhere on the instance
- it will always run from the directory:

`$FDIR/deploy/`

After a fresh clone, need to call:

```
firesim managerinit
```

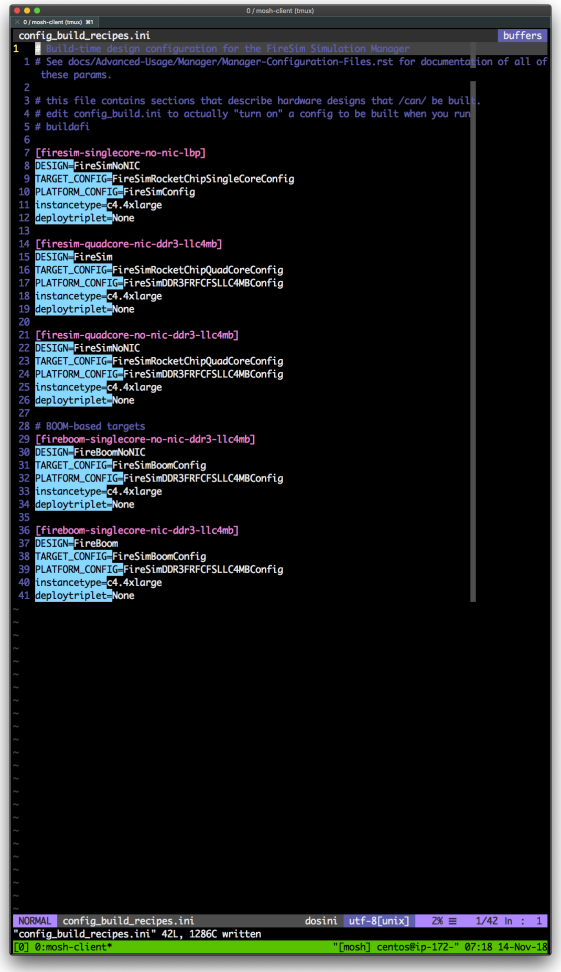
→ You did this at the start of the tutorial

Interactive:

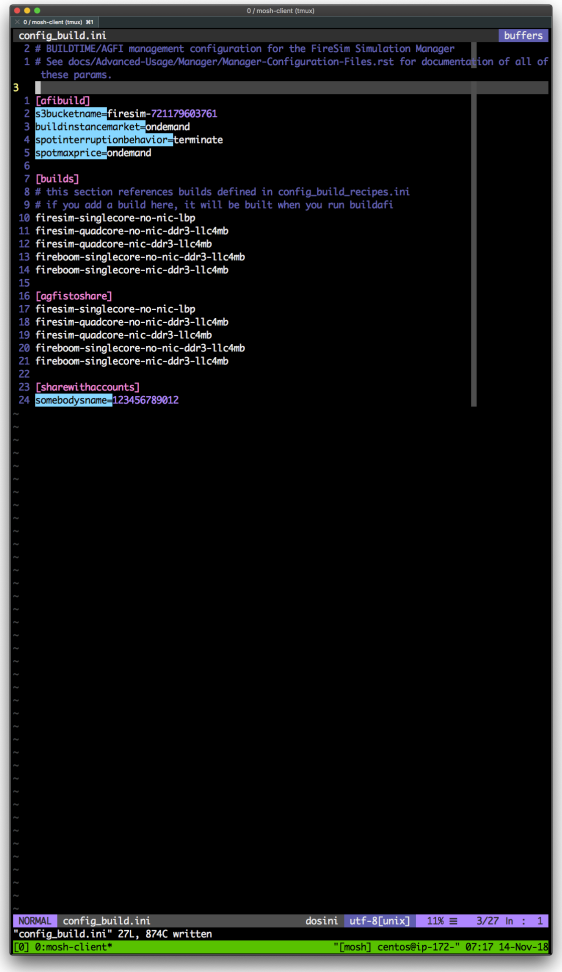
```
$ cd $FDIR/deploy

$ ls
```

Configuring the Manager. 4 files in `firesim/deploy/`

`config_build_recipes.ini`

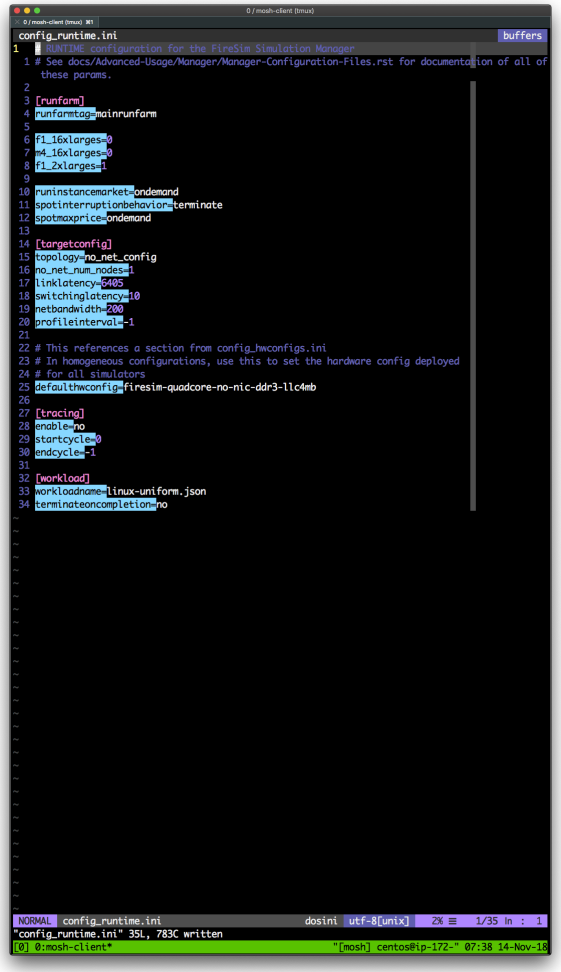


`config_build.ini`



`config_hwdb.ini`

`config_runtime.ini`



Configuring a Build

0 / mosh-client (tmux)

0 / mosh-client (tmux) ⌘1

config_build_recipes.ini buffers

```
# Build-time design configuration for the FireSim Simulation Manager
# See docs/Advanced-Usage/Manager/Manager-Configuration-Files.rst for documentation of all of these params.

# this file contains sections that describe hardware designs that /can/ be built.
# edit config_build.ini to actually "turn on" a config to be built when you run
# buildafi

[firesim-singlecore-no-nic-lbp]
DESIGN=FireSimNoNIC
TARGET_CONFIG=FireSimRocketChipSingleCoreConfig
PLATFORM_CONFIG=FireSimConfig
instancetype=c4.4xlarge
deploytriplet=None

[firesim-quadcore-nic-ddr3-llc4mb]
DESIGN=FireSim
TARGET_CONFIG=FireSimRocketChipQuadCoreConfig
PLATFORM_CONFIG=FireSimDDR3FRFCFSLLC4MBConfig
instancetype=c4.4xlarge
deploytriplet=None

[firesim-quadcore-no-nic-ddr3-llc4mb]
DESIGN=FireSimNoNIC
TARGET_CONFIG=FireSimRocketChipQuadCoreConfig
PLATFORM_CONFIG=FireSimDDR3FRFCFSLLC4MBConfig
instancetype=c4.4xlarge
deploytriplet=None

# BOOM-based targets
[fireboom-singlecore-no-nic-ddr3-llc4mb]
DESIGN=FireBoomNoNIC
TARGET_CONFIG=FireSimBoomConfig
PLATFORM_CONFIG=FireSimDDR3FRFCFSLLC4MBConfig
instancetype=c4.4xlarge
deploytriplet=None

[fireboom-singlecore-nic-ddr3-llc4mb]
DESIGN=FireBoom
TARGET_CONFIG=FireSimBoomConfig
```

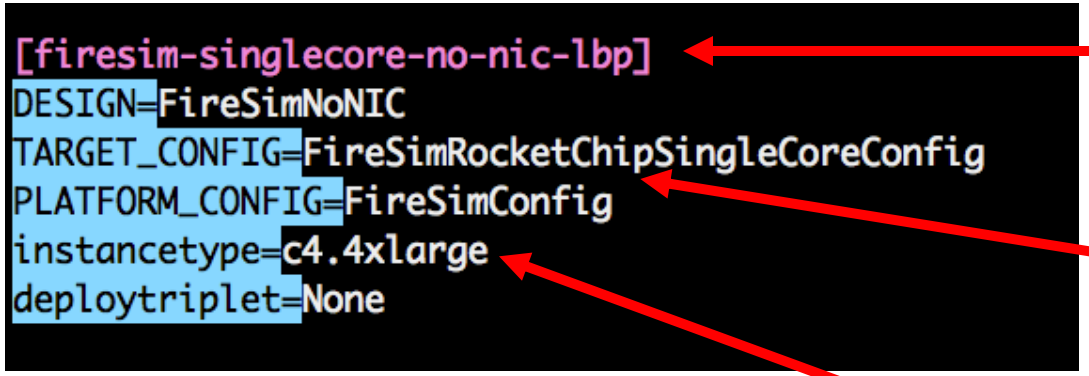
0 / mosh-client (tmux)

0 / mosh-client (tmux) ⌘1

config_build.ini buffers

```
# BUILDTIME/AGFI management configuration for the FireSim Simulation Manager
# See docs/Advanced-Usage/Manager/Manager-Configuration-Files.rst for documentation of all of these params.

[afibuild]
s3bucketname=firesim-721179603761
buildinstancemarket=ondemand
spotinterruptionbehavior=terminate
spotmaxprice=ondemand

[builds]
# this section references builds defined in config_build_recipes.ini
# if you add a build here, it will be built when you run buildafi
firesim-singlecore-no-nic-lbp
firesim-quadcore-no-nic-ddr3-llc4mb
firesim-quadcore-nic-ddr3-llc4mb
fireboom-singlecore-no-nic-ddr3-llc4mb
fireboom-singlecore-nic-ddr3-llc4mb

[agfistoshare]
firesim-singlecore-no-nic-lbp
firesim-quadcore-no-nic-ddr3-llc4mb
firesim-quadcore-nic-ddr3-llc4mb
fireboom-singlecore-no-nic-ddr3-llc4mb
fireboom-singlecore-nic-ddr3-llc4mb

[sharewithaccounts]
somebodysname=123456789012
```

Anatomy of a Build Recipe



Consists of:

- A label
- The tuple from before
- The EC2 instance type you'd like to build on

Defining a Build Job: config_build.ini

```
[afibuild]
s3bucketname=firesim-721179603761
buildinstancemarket=ondemand
spotinterruptionbehavior=terminate
spotmaxprice=ondemand

[builds]
# this section references builds defined in config_build_recipes.ini
# if you add a build here, it will be built when you run buildafi
firesim-singlecore-no-nic-lbp
firesim-quadcore-no-nic-ddr3-llc4mb
firesim-quadcore-nic-ddr3-llc4mb
fireboom-singlecore-no-nic-ddr3-llc4mb
fireboom-singlecore-nic-ddr3-llc4mb

[agfistoshare]
firesim-singlecore-no-nic-lbp
firesim-quadcore-no-nic-ddr3-llc4mb
firesim-quadcore-nic-ddr3-llc4mb
fireboom-singlecore-no-nic-ddr3-llc4mb
fireboom-singlecore-nic-ddr3-llc4mb

[sharewithaccounts]
somebodysname=123456789012
```

Consists of:

- More instance configurations
- A list of recipes you'd like to batch out to a build farm

Once you're done with builds:

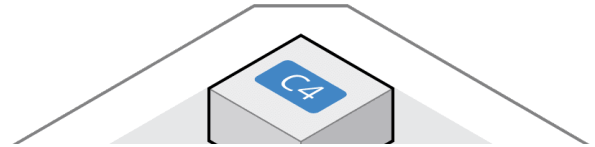
- A list of recipes you'd like to share with other users

Running builds

- Once we've configured ***what*** we want to build, let's build it

```
$ firesim buildafi
```

- This completely automates the process. For each design, in-parallel:
  - Launch a build instance (c5.4xlarge)
  - Generate target RTL & invokes Golden Gate
  - Ship infrastructure to build instances, run Vivado FPGA builds in parallel
  - Collect results back onto manager instance
    - $FDIR/deploy/results-build/<TIMESTAMP>-<tuple>/
  - Email you the entry to put into config_hwdb.ini
  - Terminate the build instance

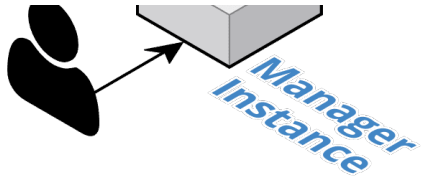


AWS Notifications <no-reply@sns.amazonaws.com>

to me

Your AGFI has been created!
Add
[firesim-singlecore-sha3-l2-no-nic-ddr3-llc4mb]
agfi=agfi-0679d5d17ba885886
deploytripletoverride=None
customruntimeconfig=None

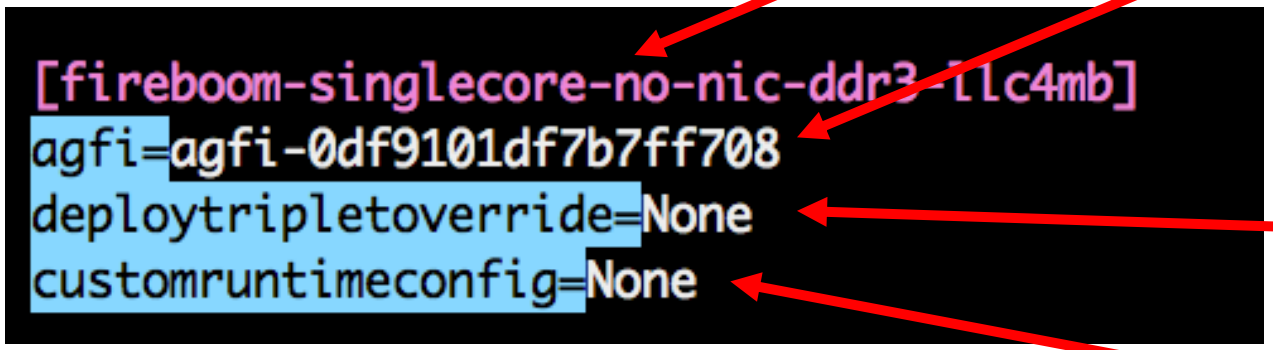
to your config_hwdb.ini to use this hardware configuration.



Interactive:

```
$ cd $FDIR/deploy

# Should print the FPGA image from the AM

$ cat built-hwdb-entries/*
```

Anatomy of a HWDB Entry



- Same label as before
- The FPGA image

Hooks to change:

- Software models
- Runtime arguments

→ *Without FPGA recompilation*

Interactive:

```
# Prefetching for the next section

$ cd ~/chipyard-afternoon/generators/sha3/software/

$ marshal -d \
    build marshal-configs/sha3-linux-test.yaml
```

Summary

- Don’t fret if you didn’t catch everything, everything we showed you today is documented in excruciating detail at http://docs.fires.im
- We learned how to:
  - Build FireSim FPGA images for a set of targets
    - http://docs.fires.im/en/latest/Building-a-FireSim-AFI.html