FireSim

Modifying the Target Design

FireSim Intensive
Chisel Community Conference 2018
Speaker: David Biancolin

Berkeley Architecture Research

Modifying the Target Design

In this section:

- Explain how the target is modeled in FireSim
- Discuss how to add and modify each class of model

Code references apply to FireSim v.1.4.0

Relevant FireSim docs section: “Targets”

https://docs.fires.im/en/latest/Advanced-Usage/Generating-Different-Targets.html

Berkeley Architecture Research

Important Submodules & Aliases

Submodules:

MIDAS:	`sim/midas`
FIRRTL:	`sim/firrtl`
Firechip:	`target-design/firechip`
Rocket-Chip:	`target-design/firechip/rocket-chip`
Chisel:	`target-design/firechip/rocket-chip/chisel`

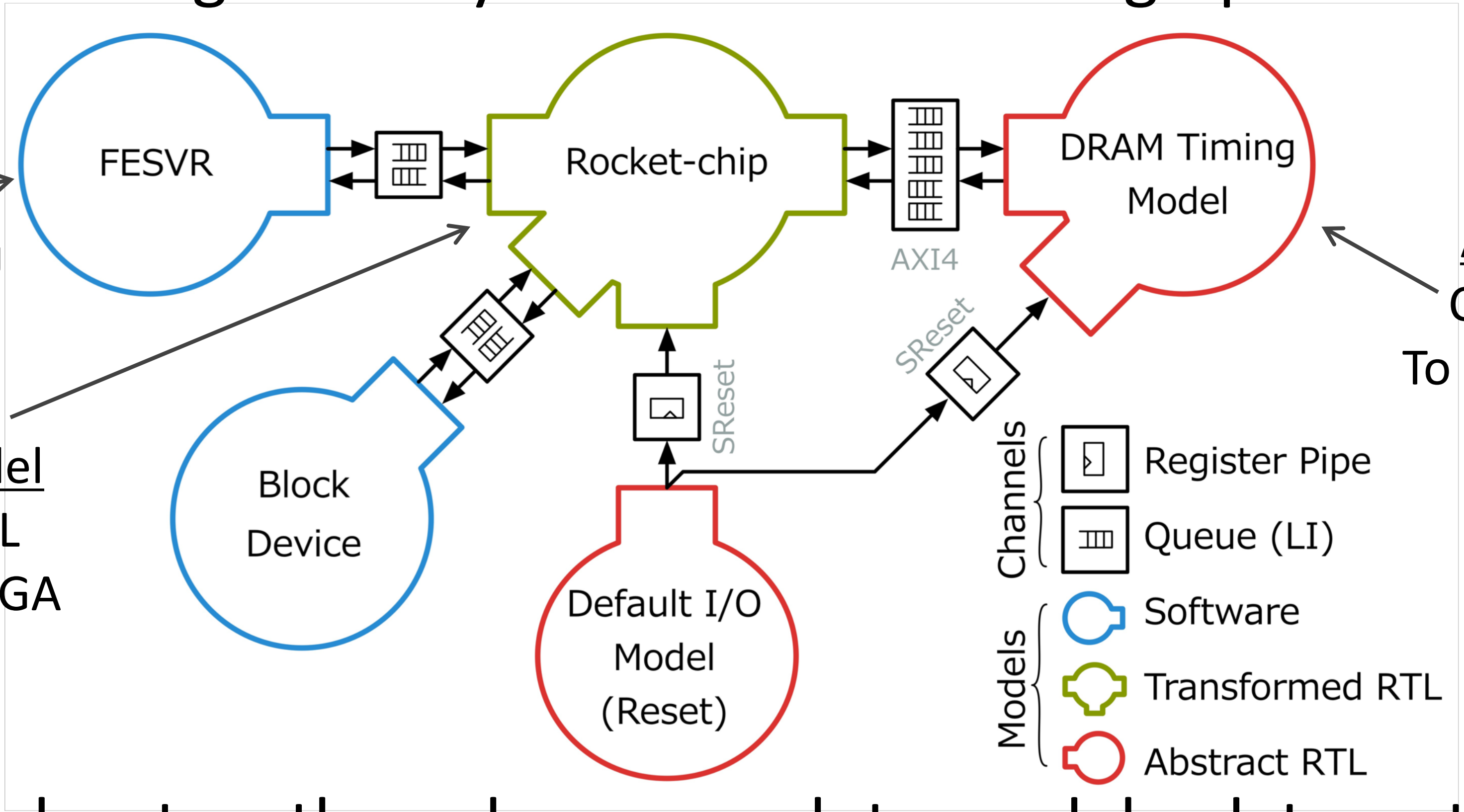
Aliases for this presentation:

`FSCALA`	-> sim/src/main/scala
`FCPP`	-> sim/src/main/cc/
`MSCALA`	-> {in MIDAS}/src/main/scala/midas
`MCPP`	-> {in MIDAS}/src/main/cc/

Berkeley Architecture Research

From Yesterday: Target as a Dataflow Graph

- Represent the target as a synchronous-dataflow graph



SW Model
To be hosted on a CPU

Transformed RTL Model
Derived from SoC RTL
To be hosted on an FPGA

Abstract RTL Model
Cannot be taped out
To be hosted on an FPGA

- FireSim stitches together a larger graph to model a datacenter

Berkeley Architecture Research

Expressing the Target Graph in FireSim

- Target graph is implicit, currently not encoded in one place.
- There is only one transformed-RTL model
  - The CHIRRTL you passed to midas.Compiler
- In Simulation Mapping
  - 2+ entry queue channels are instantiated on Decoupled IO
  - 1-cycle pipe channels are used everywhere else.
- SW and Custom RTL models are bound to IO bundles using `midas.core.Endpoint`
  - Matches on a IO type, generates a model or endpoint

Parameterizing the RTL-Transformed Model

- FireSim provides four base Rocket-Chip “cakes” that form the base RTL-transformed model
  - `DESIGN` make variable

Two types of parameters:

1. Target: (see TargetConfigs.scala) Configure the generated RTL model
   - # of Cores, L1 $ Sizes, # of memory channels
   - `TARGET_CONFIG` make variable
2. Sim/Platform: (see SimConfigs.scala) Configure MIDAS, SW/Custom RTL model generation
   - Configs for endpoints, Type of DRAM model, enable debug features etc…
   - `PLATFORM_CONFIG` make variable

Swapping out the RTL-Transformed Model

Two common approaches, based on the type of Target:

1. Project-template-based Rocket-chip targets (FireChip)
   - Point at the FireChip submodule at your fork

   NB: MIDAS depends on Rocket-Chip; Rocket-Chip provides Chisel submodule
2. Everything else (ex. Your custom CGRA etc..)
   - Add scala sources for your target (modify `sim/build.sbt`)
   - Define an App (Generator) that calls `midas.Compiler(…)`
   - Add a target-specific makefrag (ex. `sim/src/main/makefrag/firesim/Makefrag`)

   See the MIDAS examples for an simple demonstration of this.

On Abstract RTL & SW Models.

FireSim & MIDAS provide the following models.

Abstract-RTL :

1. DefaultIOModel (`$MSCALA/widgets/PeekPokeIO.scala`)
   - Bound to I/O unbound by custom endpoints
2. DRAM & LLC model (`$MSCALA/models/dram/MidasMemModel.scala`)
   - Bound to AXI4 slave interfaces

SW-models:

1. NIC
2. SerialIO
3. Block-device
4. UART

These are all FireSim provided: See `$FSCALA/endpoints` and `$FCPP/endpoints`

Compile-time vs Runtime Configuration

Software models and custom RTL models are configured twice:

1. Compile/Generation time
2. Runtime

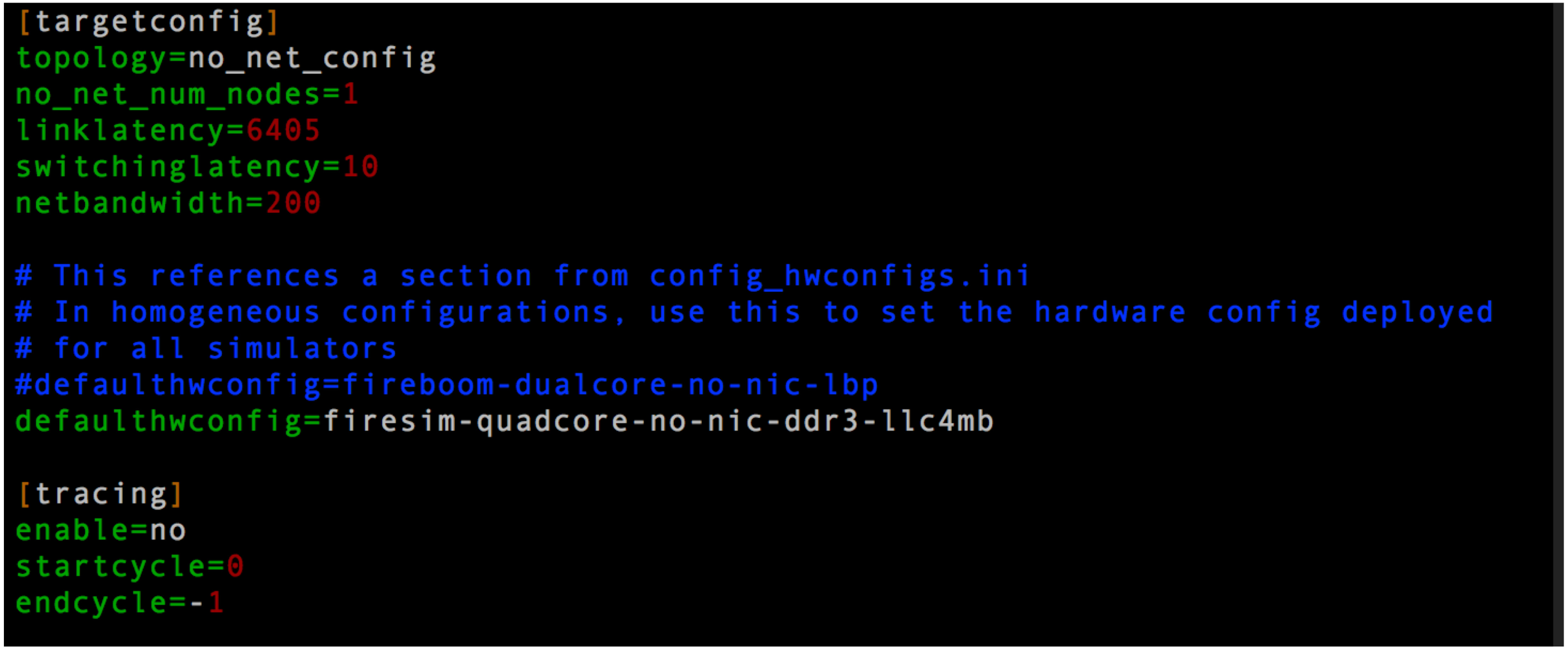
On Compile-time configuration:

- RTL-generation for Custom-RTL models (FPGA resynthesis required)
    - How: using different a `PLATFORM_CONFIG`; hacking on the Chisel sources
- CPP compilation for SW models (no FPGA resynthesis)
    - How: changing model & driver CPP sources.

Runtime Configuration

- Pass plus args to the simulator
  - SW models: does what you expect
  - Custom-RTL models: driver writes to model's configuration registers
  - Ex. +blkdev_write_latency,
- How: Modify `deploy/config_runtime.ini`
  - Change ini-exposed variables
  - Specify a customruntimeconfig in `deploy/config_hwdb.ini`

Runtime Configuration – config_runtime.ini



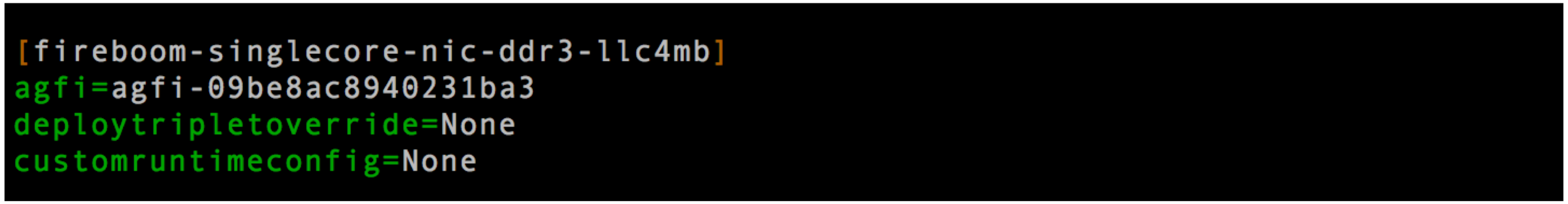
```
[targetconfig]
topology=no_net_config
no_net_num_nodes=1
linklatency=6405
switchinglatency=10
netbandwidth=200

# This references a section from config_hwconfigs.ini
# In homogeneous configurations, use this to set the hardware config deployed
# for all simulators
#defaulthwconfig=fireboom-dualcore-no-nic-lbp
defaulthwconfig=firesim-quadcore-no-nic-ddr3-llc4mb

[tracing]
enable=no
startcycle=0
endcycle=-1
```

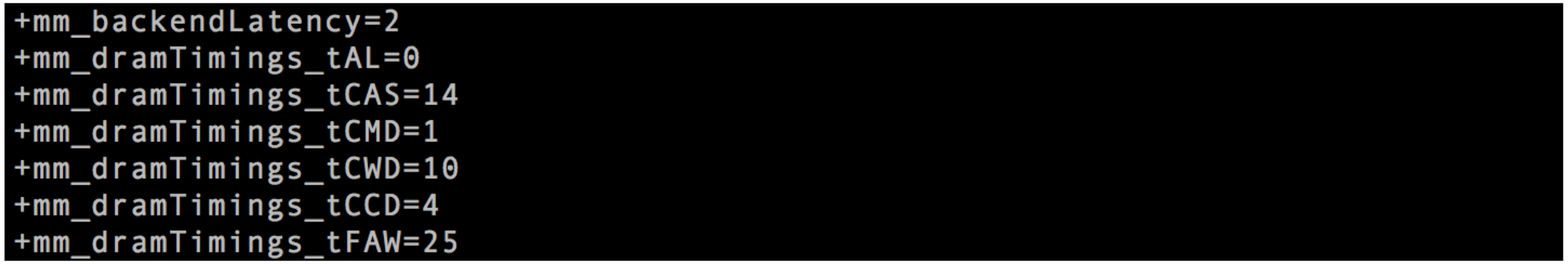
Berkeley Architecture Research

Runtime Configuration – config_hwdb.ini



```
[fireboom-singlecore-nic-ddr3-llc4mb]
agfi=agfi-09be8ac8940231ba3
deploytripletoverride=None
customruntimeconfig=None
```

HWDB entry: None -> Uses a default runtime.conf emitted at generation time



```
+mm_backendLatency=2
+mm_dramTimings_tAL=0
+mm_dramTimings_tCAS=14
+mm_dramTimings_tCMD=1
+mm_dramTimings_tCWD=10
+mm_dramTimings_tCCD=4
+mm_dramTimings_tFAW=25
```

Runtime.conf -> plusArgs you want to pass to the simulator

Adding a Custom Model – Chisel-side

1. Expose a Record/Bundle in your Target RTL's IO with a specific type
2. Extend midas.Endpoint to:
    1. Match on the correct Chisel type
        - Implement `Endpoint.matchType(Data)=> Boolean`
    2. Generate your SW model's endpoint, or custom RTL-model Implement
        - Implement `Endpoint.widget(Parameters)=> EndpointWidget`
3. Add your endpoint to the EndpointMap Field in your `PLATFORM_CONFIG`

See:

- $FSCALA/endpoints/UARTWidget.scala
- $FSCALA/endpoints/BlockDevWidget.scala
- $FSCALA/firesim/SimConfigs.scala

Adding a Custom Model – Driver-side

1. Define an endpoint driver class
   - Use simif::{read, write, push, pull} to interact with FPGA-hosted endpoint
2. Register the endpoint driver in firesim_top.cc

See:

- $FCPP/endpoints/uart.{cc, h}
- $FCPP/endpoints/blockdev.{cc, h}
- $FCPP/firesim/firesim_top.cc

DRAM Timing Models – Generation Time Conf

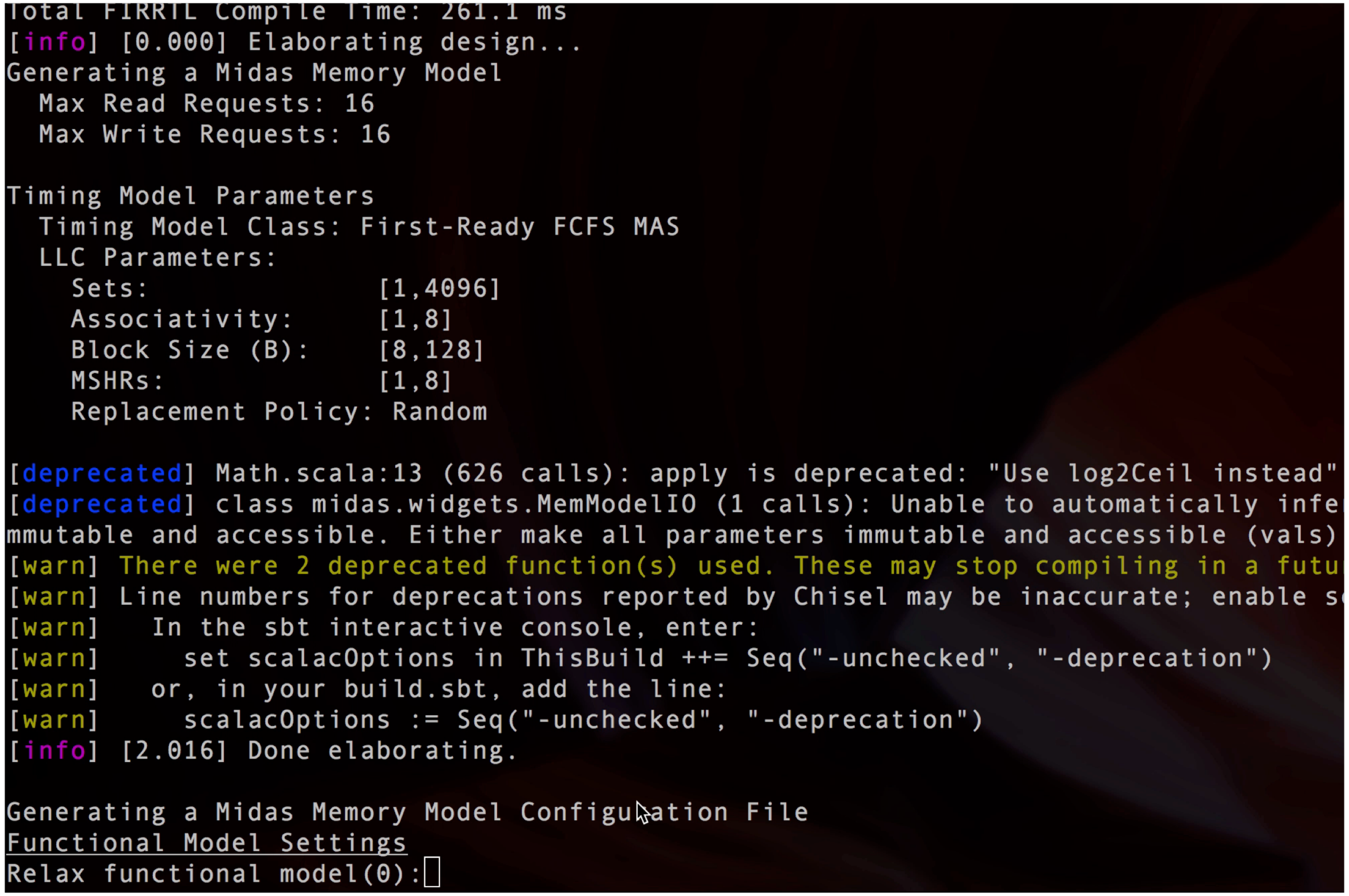
- Parameterized using the PLATFORM_CONFIG make variable
  - See `$FSCALA/firesim/SimConfigs.scala`
- Abstract timing models:
  - Latency-Bandwidth Pipe
  - Bank Conflict
- DDR3 Timing models:
  - First-Come First-Served (FCFS)
  - First-Ready, First-Come First-Served (FR-FCFS)
- All timing models can be composed with a LLC model.

DRAM Timing Models – Runtime Conf

- DDR timing models expose >30 runtime arguments
  - Many illegal settings, need to validate against generated model instance
    - Ex. Setting that would overflow a timing register.
  - Many invalid settings, need to validate against DDR3 part database
    - Ex. A non-standard JEDEC DDR3 timing, timings are density dependent
- A default runtime-configuration is emitted at generation time
  - See `generated-src/f1/${TARGET_TUPLE}/runtime.conf`
- To generate a custom runtime-config:
  - In sim/: `make conf`

Berkeley Architecture Research

DRAM Timing Models – custom runtime.conf



```
Total FIRRTL Compile Time: 261.1 ms
[info] [0.000] Elaborating design...
Generating a Midas Memory Model
  Max Read Requests: 16
  Max Write Requests: 16

Timing Model Parameters
  Timing Model Class: First-Ready FCFS MAS
  LLC Parameters:
    Sets:               [1,4096]
    Associativity:      [1,8]
    Block Size (B):     [8,128]
    MSHRs:              [1,8]
    Replacement Policy: Random

[deprecated] Math.scala:13 (626 calls): apply is deprecated: "Use log2Ceil instead"
[deprecated] class midas.widgets.MemModelIO (1 calls): Unable to automatically infe
mmutable and accessible. Either make all parameters immutable and accessible (vals)
[warn] There were 2 deprecated function(s) used. These may stop compiling in a futu
[warn] Line numbers for deprecations reported by Chisel may be inaccurate; enable s
[warn]   In the sbt interactive console, enter:
[warn]     set scalacOptions in ThisBuild ++= Seq("-unchecked", "-deprecation")
[warn]   or, in your build.sbt, add the line:
[warn]     scalacOptions := Seq("-unchecked", "-deprecation")
[info] [2.016] Done elaborating.

Generating a Midas Memory Model Configuration File
Functional Model Settings
Relax functional model(0):
```