

FPGA Prototyping

Nayiri Krzysztofowicz

UC Berkeley

nayiri@berkeley.edu



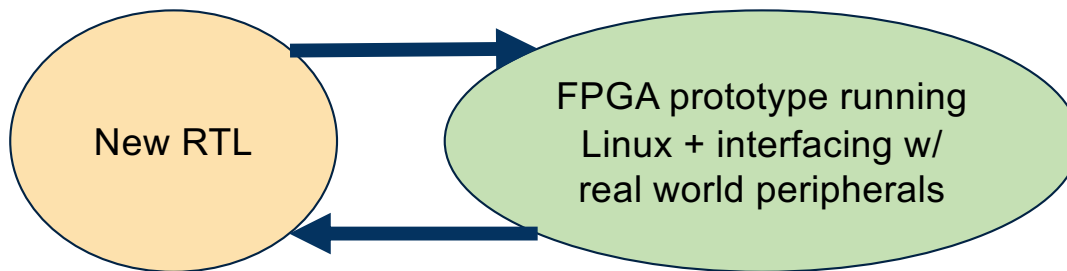
Berkeley
Architecture
Research

CHIP  **YARD**

Motivation



- FPGAs are a powerful tool!
- Many people have off the shelf FPGAs
- Chipyard use cases for FPGAs
 - HW simulation – Use FireSim!
 - Interact with the real-world peripherals
 - Tapeout bringup platform



Goals



- Understand the basics of generating a bitstream
- Overview of the two supported platforms
- Example: Build and run Linux on a VCU118 FPGA w/ BOOM!
 - Build a pre-configured BOOM bitstream
 - Build Linux binary and start the prototype run
 - Interact with Linux!



How things will work



```
# command 1
> echo "Chipyard Rules!"

# command 2
> do_this arg1 arg2
```

Terminal Section

```
// SOME COMMENT HERE
class SmallBoomConfig extends Config(
  new WithTop ++
  new WithBootROM ++
  new boom.common.WithSmallBooms ++
  new boom.common.WithNBoomCores(1) ++
  new freechips.rocketchip.system.BaseConfig)
```

Inside-a-File Section





Getting Started



Prerequisites



- Vivado installed and on your PATH
 - Tested with 2018.3 but should work for some other versions
- Fully setup Chipyard
 - All submodules initialized
 - A toolchain installed
- Basic understanding of Vivado
 - How to load a bitstream
 - How to connect to an FPGA



Example



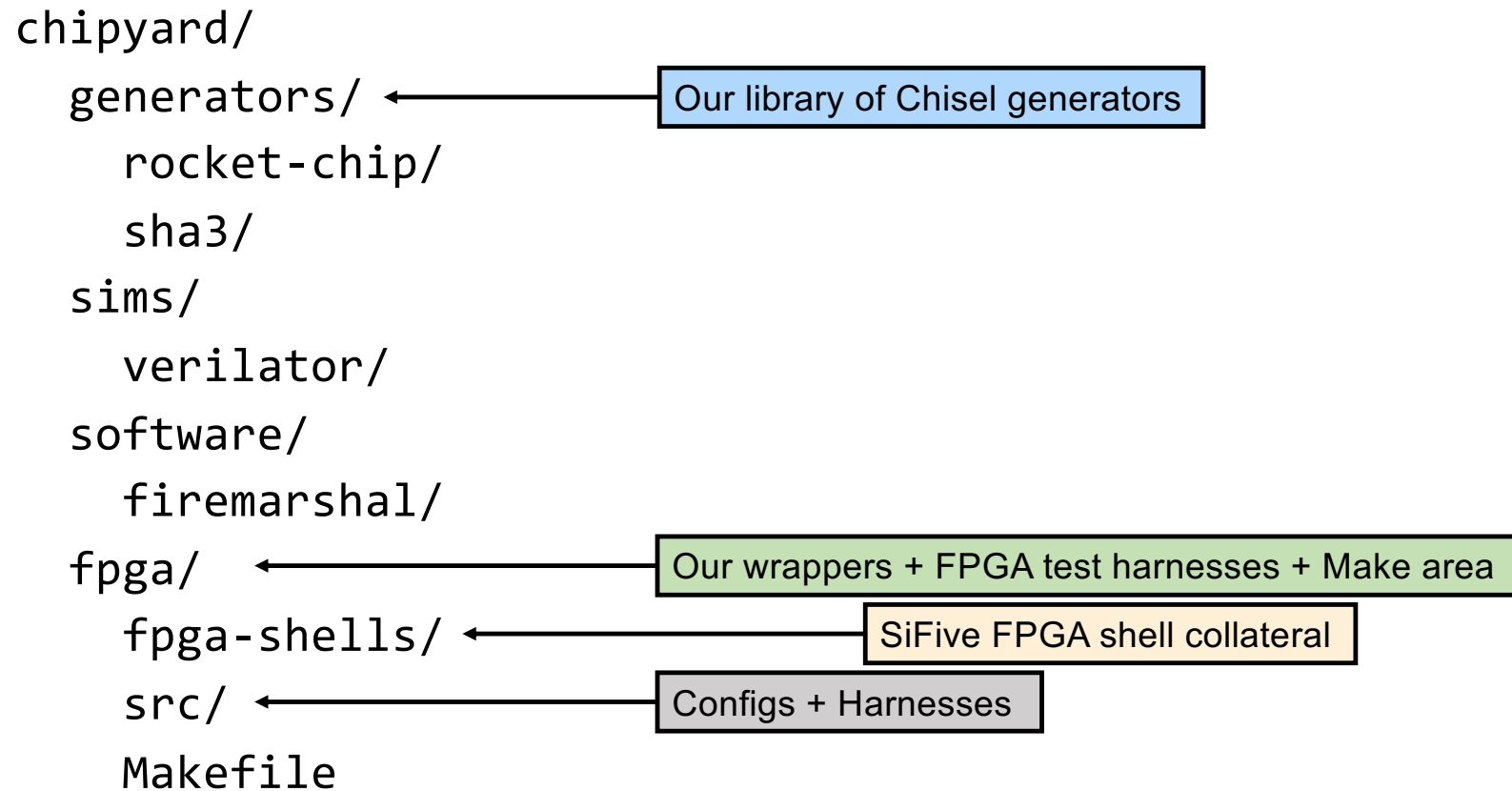
```
# return to Chipyard
> cd ~/chipyard
> ls

# setup the repo to build fpga images
> ./scripts/init-fpga.sh
```

Wrapper around `git submodule init`
to clone the necessary FPGA
collateral



Directory Structure



FPGA-Shells



- Provided by SiFive
 - But extended and built upon by UCB students
 - Used in the SiFive Freedom platform
- Connects Rocket Chip SoCs to FPGAs
- Contains board and peripheral support
- Current FPGA support
 - VCU118
 - Arty A7

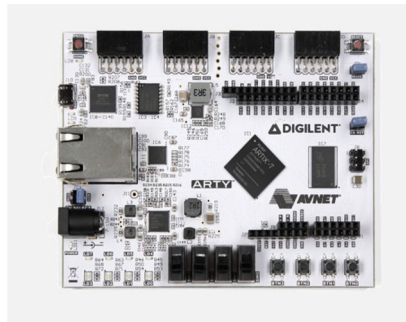
```
chipyard/  
  generators/  
    rocket-chip/  
      boom/  
      sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
    src/  
  tools/  
    chisel/  
    firrtl/  
  tests/  
  build.sbt
```



Chipyard supported boards



Xilinx Arty



- Artix7 FPGA fits TinyRocketConfig
- UART, JTAG, QSPI flash for BootROM
- No backing DDR (yet), so does not boot Linux

Xilinx VCU118

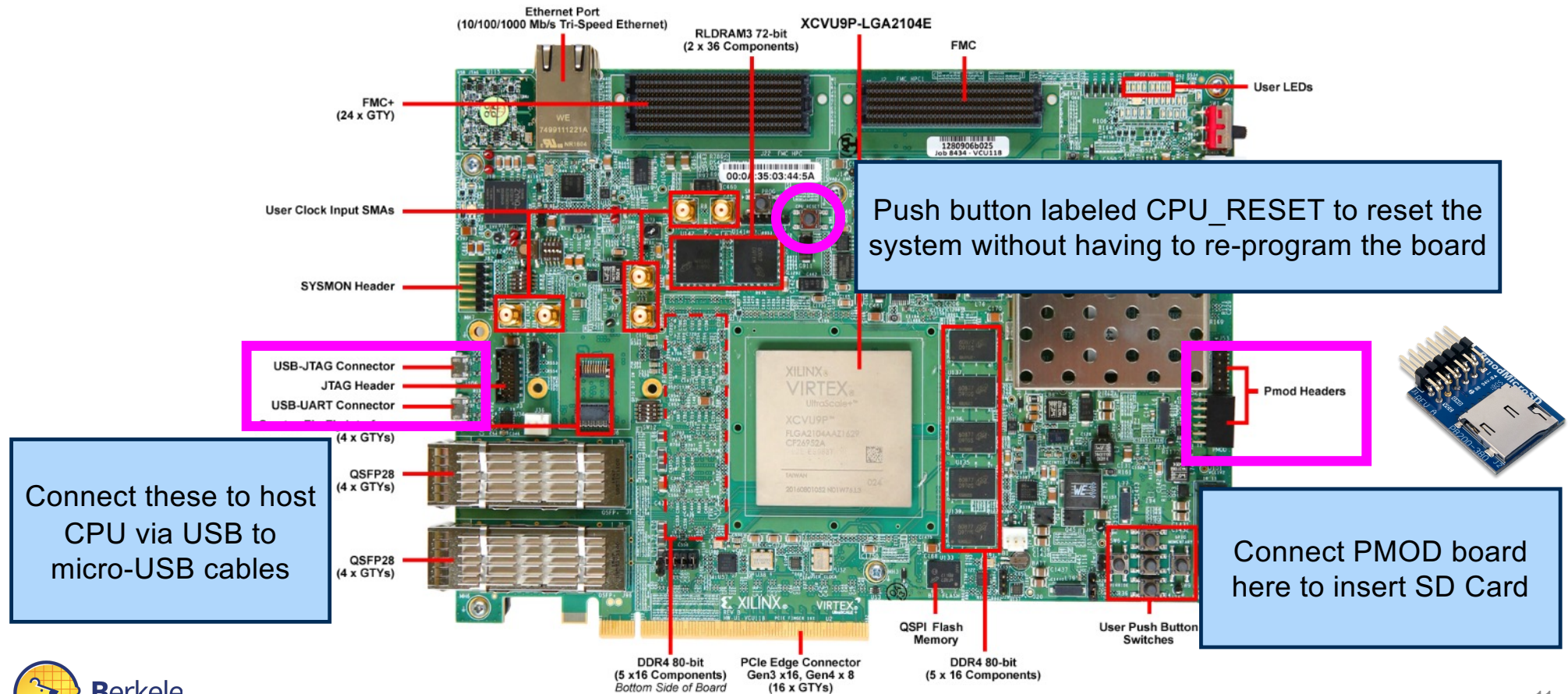


- Ultrascale FPGA fits large Rocket and BOOM configs
 - UART, JTAG, DDR backing memory, SDCard boot BootROM
 - Boots Linux

```
chipyard/  
  generators/  
    rocket-chip/  
    boom/  
    sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
    src/  
  tools/  
    chisel/  
    firrtl/  
  tests/  
  build.sbt
```



Navigating the VCU118 Board





Build a VCU118 bitstream



Bitstreams



- Building a bitstream is similar to building a simulator binary in `sims/*` but in `fpga/`
 - Converts Chisel to Verilog
 - Runs Verilog through Vivado to create a bitstream
- Target a specific configuration + FPGA

```
# build the BOOM config bitstream for VCU118
> make SUB_PROJECT=vcu118 CONFIG=BoomVCU118Config
bitstream
```

```
chipyard/
  generators/
    rocket-chip/
      boom/
      sha3/
  sims/
    verilator/
    fpga/
      fpga-shells/
      src/
  tools/
    chisel/
    firrtl/
  tests/
  build.sbt
```



Bitstreams



- Building a bitstream is similar to building a simulator binary in `sims/*` but in `fpga/`
 - Converts Chisel to Verilog
 - Runs Verilog through Vivado to create a bitstream
- Target a specific configuration + FPGA

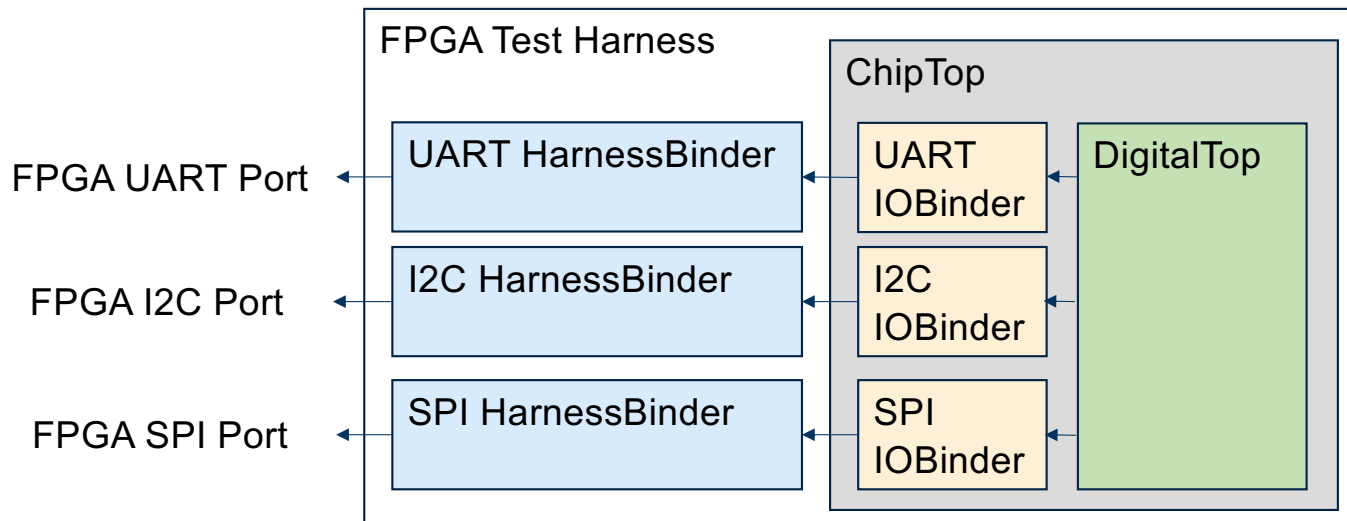
```
# build the BOOM config bitstream for VCU118
> make SUB_PROJECT=vcu118 CONFIG=BoomVCU118Config
bitstream
```

This will take a loooong time! It is generating the Verilog, and passing it to Vivado to create the bitstream

```
chipyard/
  generators/
    rocket-chip/
      boom/
      sha3/
    sims/
      verilator/
      fpga/
      fpga-shells/
      src/
    tools/
      chisel/
      firrtl/
    tests/
  build.sbt
```



Anatomy of an FPGA prototype



```
chipyard/  
  generators/  
    rocket-chip/  
      boom/  
        sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
      src/main/scala/vcu118/  
        Configs.scala  
  tools/  
    chisel/  
    firrtl/  
  tests/  
  build.sbt
```

- HarnessBinders connect ChipTop IOs to FPGA specific ports given by FPGA-Shells
- TestHarness is associated with FPGA platform

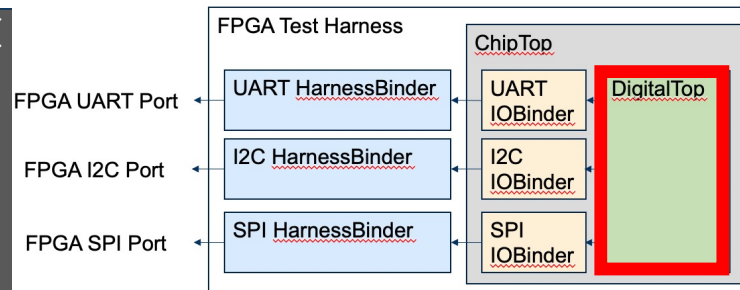


View of the config



```
class WithVCU118Tweaks extends Config(  
  new WithUART ++  
  new WithSPISDCard ++  
  new WithDDRMem ++  
  new WithUARTIOPassthrough ++  
  new WithSPIIOPassthrough ++  
  new WithTLIOPassthrough ++  
  new WithDefaultPeripherals ++  
  new chipyard.config.WithTLBackingMemory ++  
  new WithSystemModifications ++  
  new chipyard.config.WithNoDebug ++  
  new freechips.rocketchip.subsystem.WithoutTLMonitors ++  
  new freechips.rocketchip.subsystem.WithNMemoryChannels(1))
```

```
class BoomVCU118Config extends Config(  
  new WithFPGAFrequency(50) ++  
  new WithVCU118Tweaks ++  
  new chipyard.MegaBoomConfig)
```



Default Chipyard configuration

```
chipyard/  
  generators/  
    rocket-chip/  
    boom/  
    sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
    src/main/scala/vcu118/  
      Configs.scala  
  tools/  
    chisel/  
    firrtl/  
  tests/  
  build.sbt
```



View of the config



```
class WithVCU118Tweaks extends Config(  
  new WithUART ++  
  new WithSPISDCard ++  
  new WithDDRMem ++  
  new WithUARTIOPassthrough ++  
  new WithSPIIOPassthrough ++  
  new WithTLIOPassthrough ++  
  new WithDefaultPeripherals ++  
  new chipyard.config.WithTLBackingMemory ++  
  new WithSystemModifications ++  
  new chipyard.config.WithNoDebug ++  
  new freechips.rocketchip.subsystem.WithoutTLMonitors ++  
  new freechips.rocketchip.subsystem.WithNMemoryChannels(1))
```

```
class BoomVCU118Config extends Config(  
  new WithFPGAFrequency(50) ++  
  new WithVCU118Tweaks ++  
  new chipyard.MegaBoomConfig)
```

Set FPGA frequency in MHz

```
chipyard/  
  generators/  
    rocket-chip/  
      boom/  
        sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
    src/main/scala/vcu118/  
      Configs.scala  
  tools/  
    chisel/  
    firrtl/  
  tests/  
  build.sbt
```



View of the config



```
class WithVCU118Tweaks extends Config(  
  new WithUART ++  
  new WithSPISDCard ++  
  new WithDDRMem ++  
  new WithUARTIOPassthrough ++  
  new WithSPIIOPassthrough ++  
  new WithTLIOPassthrough ++  
  new WithDefaultPeripherals ++  
  new chipyard.config.WithTLBackingMemory ++  
  new WithSystemModifications ++  
  new chipyard.config.WithNoDebug ++  
  new freechips.rocketchip.subsystem.WithoutTLMonitors ++  
  new freechips.rocketchip.subsystem.WithNMemoryChannels(1))
```

```
class BoomVCU118Config extends Config(  
  new WithFPGAFrequency(50) ++  
  new WithVCU118Tweaks ++  
  new chipyard.MegaBoomConfig)
```

Link this configuration with the upper configuration fragment

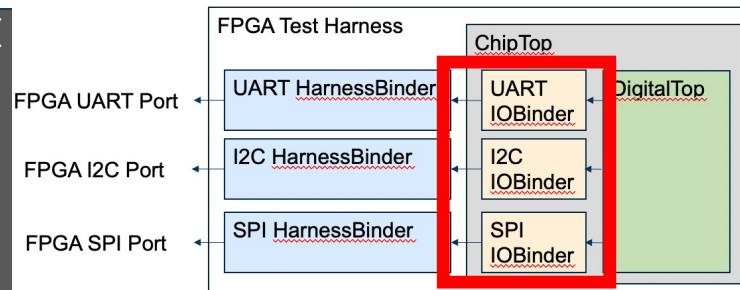
```
chipyard/  
  generators/  
    rocket-chip/  
      boom/  
        sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
    src/main/scala/vcu118/  
      Configs.scala  
  tools/  
    chisel/  
    firrtl/  
  tests/  
  build.sbt
```



View of the config



```
class WithVCU118Tweaks extends Config(  
  new WithUART ++  
  new WithSPISDCard ++  
  new WithDDRMem ++  
  new WithUARTIOPassthrough ++  
  new WithSPIIOPassthrough ++  
  new WithTLIOPassthrough ++  
  new WithDefaultPeripherals ++  
  new chipyard.config.WithTLBackingMemory ++  
  new WithSystemModifications ++  
  new chipyard.config.WithNoDebug ++  
  new freechips.rocketchip.subsystem.WithoutTLMonitors ++  
  new freechips.rocketchip.subsystem.WithNMemoryChannels(1))  
  
class BoomVCU118Config extends Config(  
  new WithFPGAFrequency(50) ++  
  new WithVCU118Tweaks ++  
  new chipyard.MegaBoomConfig)
```



Add IOBinders

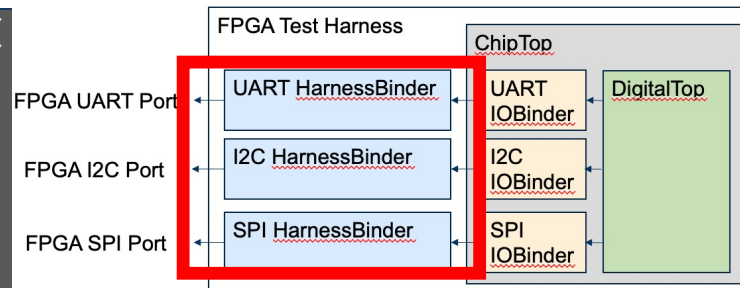
```
chipyard/  
  generators/  
    rocket-chip/  
    boom/  
    sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
    src/main/scala/vcu118/  
      Configs.scala  
  tools/  
    chisel/  
    firrtl/  
  tests/  
  build.sbt
```



View of the config



```
class WithVCU118Tweaks extends Config(  
  new WithUART ++  
  new WithSPISDCard ++  
  new WithDDRMem ++  
  new WithUARTIOPassthrough ++  
  new WithSPIIOPassthrough ++  
  new WithTLIOPassthrough ++  
  new WithDefaultPeripherals ++  
  new chipyard.config.WithTLBackingMemory ++  
  new WithSystemModifications ++  
  new chipyard.config.WithNoDebug ++  
  new freechips.rocketchip.subsystem.WithoutTLMonitors ++  
  new freechips.rocketchip.subsystem.WithNMemoryChannels(1))  
  
class BoomVCU118Config extends Config(  
  new WithFPGAFrequency(50) ++  
  new WithVCU118Tweaks ++  
  new chipyard.MegaBoomConfig)
```



Add HarnessBinders

```
chipyard/  
  generators/  
    rocket-chip/  
      boom/  
        sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
    src/main/scala/vcu118/  
      Configs.scala  
  tools/  
    chisel/  
    firrtl/  
  tests/  
  build.sbt
```



View of the config



```
class WithVCU118Tweaks extends Config(  
  new WithUART ++  
  new WithSPISDCard ++  
  new WithDDRMem ++  
  new WithUARTIOPassthrough ++  
  new WithSPIIOPassthrough ++  
  new WithTLIOPassthrough ++  
  new WithDefaultPeripherals ++  
  new chipyard.config.WithTLBackingMemory ++
```

```
  new WithSystemModifications ++
```

```
  new chipyard.config.WithNoDebug ++
```

```
  new freechips.rocketchip.subsystem.WithoutTLMonitors ++
```

```
  new freechips.rocketchip.subsystem.WithNMemoryChannels(1))
```

```
class BoomVCU118Config extends Config(  
  new WithFPGAFrequency(50) ++  
  new WithVCU118Tweaks ++  
  new chipyard.MegaBoomConfig)
```

Setup buses, use SDCard
bringup bootrom, set
memory size, and more

```
chipyard/  
  generators/  
    rocket-chip/  
      boom/  
      sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
    src/main/scala/vcu118/  
      Configs.scala  
  tools/  
    chisel/  
    firrtl/  
  tests/  
  build.sbt
```





Building Linux for FPGA prototypes



Using FireMarshal to build Linux



- Later in the tutorial we will go into more depth on FireMarshal
 - A unified workload generation tool used across Chipyard

```
# navigate to firemarshal (assuming pre-setup)
> cd chipyard/software/firemarshal

# configure firemarshal for fpga prototypes
> echo "board-dir : 'boards/prototype'" > marshal-config.yaml

# build linux with initramfs
> ./marshal -v -d build br-base.json
```

```
chipyard/
  generators/
    rocket-chip/
    boom/
    sha3/
  sims/
    verilator/
  fpga/
    fpga-shells/
    src/
  software/
    firemarshal/
  tests/
  build.sbt
```



Using FireMarshal to build Linux



- Later in the tutorial we will go into more depth on FireMarshal
 - A unified workload generation tool used across Chipyard

```
# navigate to firemarshal (assuming pre-setup)
> cd chipyard/software/firemarshal

# configure firemarshal for fpga prototypes
> echo "board-dir : 'boards/prototype'" > marshal-config.yaml

# build linux with initramfs
> ./marshal -v -d build br-base.json

> ls images/
```

All collateral will be located in the
`images/` area of the `firemarshal/`
directory

```
chipyard/
  generators/
    rocket-chip/
    boom/
    sha3/
  sims/
    verilator/
  fpga/
    fpga-shells/
    src/
  software/
    firemarshal/
  sts/
  build.sbt
```

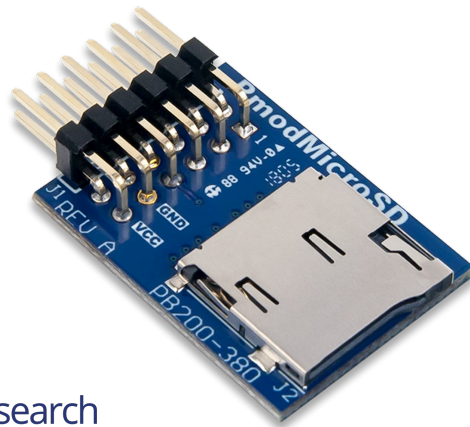


Putting Linux onto VCU118 SDCard



- By default the VCU118 platform loads binaries from a PMOD SDCard
- We need to flatten (i.e. remove the DRAM offset) of the output binary before loading it into SDCard

```
# flatten output linux binary to load on SDCard  
> ./marshal -v -d install -t prototype br-base.json
```



```
chipyard/  
  generators/  
    rocket-chip/  
      boom/  
        sha3/  
      sims/  
        verilator/  
      fpga/  
        fpga-shells/  
        src/  
      software/  
        firemarshal/  
      tests/  
      build.sbt
```



Putting Linux onto VCU118 SDCard



- Next, we need to put the binary onto a pre-setup SDCard
 - Contains 2 partitions; one to store the binary with Linux, one to store a filesystem to access from the DUT
 - SDCard setup instructions:
chipyard.readthedocs.io/en/1.5.0/Prototyping/VCU118.html#setting-up-the-sdcard

```
# move flattened binary to SDCard 1st partition (/dev/sdc1 an ex)
> sudo dd if=$PWD/images/br-base-bin-nodisk-flat of=/dev/sdc1
```

```
chipyard/
  generators/
    rocket-chip/
    boom/
    sha3/
  sims/
    verilator/
  fpga/
    fpga-shells/
    src/
  software/
    firemarshal/
  tests/
  build.sbt
```





Programming the FPGA and Running Linux



Last steps



- Use Vivado to program the FPGA with the bitstream
 - Either can use GUI or CLI
 - Bitstream - fpga/generated-src/<NAME>/obj/*.bit
- Plug in the SD Card and connect to the serial port
 - Might need to reset the softcore using CPU_RESET button

```
# connect to serial port (in this case called ttyUSB1)  
> screen -S FPGA_UART_CONSOLE /dev/ttyUSB1 115200
```

```
chipyard/  
  generators/  
    rocket-chip/  
    boom/  
    sha3/  
  sims/  
    verilator/  
  fpga/  
    fpga-shells/  
    src/  
      generated-src/  
  software/  
    firemarshal/  
  tests/  
  build.sbt
```





That's it! Demo time!

```
[07:03 PM]:[abe.gonzalez@bwrcrds1-1]:[chipyard]:[master]$
```

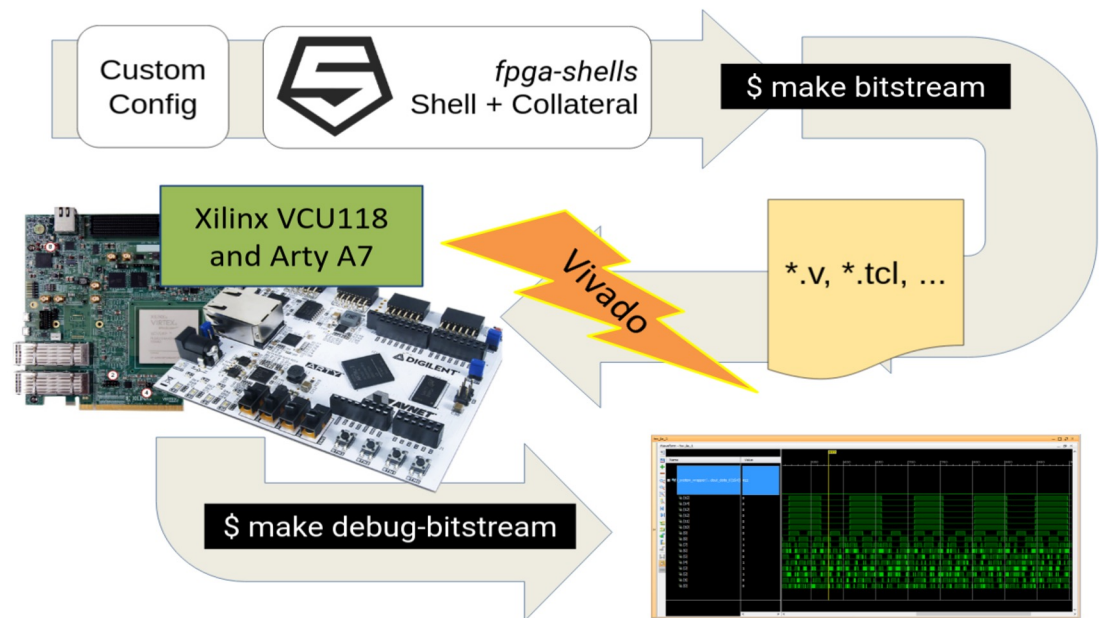
```
[isa_20210:back*
```

```
"hwcrds1-1-005-benke" 10:03:16 Jun 21
```

Conclusion



- Future Updates
 - More FPGAs supported
 - Full behavior simulation of prototype at board level
 - More peripherals
 - Better SW support
- Used internally at Berkeley
 - Real world interaction
 - Bringup Platform
 - Education
 - Outreach!





Coming up...

FireSim Introduction