Putting Scaling in Perspective

Lisa Su, HotChips'19 keynote

Performance gains over the past decade
Power and Performance Trends

What do we do next?

Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp
Cost Of Developing New Products

- These are non-recurring (NRE) costs, need to be amortized over the lifetime of a product
- We will attempt to dismantle this…

Source: IBS

Nikolić, Spring 2024
Major Roadblocks

1. Managing complexity
   How to design a 100 billion transistor chip?
   And what to use all these transistors for?

2. Cost of integrated circuits is increasing
   It takes $10M to design a chip
   Mask costs are many $M in 5nm technology
   Wafer costs are increasing (~20k)

3. Power as a limiting factor
   End of frequency scaling
   Dealing with power, leakages

4. Robustness issues
   Variations, SRAM, memory, soft errors, signal integrity

5. The interconnect problem
Assigned Reading

On an SoC generator


On transistor models (in about 2 weeks):

  • Just the scaling principles


Lecture Outline

• SoC generator: Chipyard
  • Great for class (and other) projects
RISC-V Processor Core
• Everyone has seen (or possibly designed) a 5-stage RISC-V (RV32I) core
• Needs memory (cache/scratchpad), co-processors, peripherals
**Rocket In-Order Core**

- First open-source RISC-V CPU
  - Designed as a Chisel generator

- In-order, single-issue RV64GC core
  - Floating-point via Berkeley hardfloat library
  - RISC-V Compressed
  - Physical Memory Protection (PMP) standard
  - Supervisor ISA and Virtual Memory

- Boots Linux

- Supports Rocket Chip Coprocessor (RoCC) interface

- L1 I$ and D$
  - Caches can be configured as scratchpads
Inside Rocket

- Front-end
  - Fetches instructions from I$
Rocket Pipeline

• 1x integer ALU/IMUL/IDIV + optional FPU
Data Cache

- L1 cache
To Build an SoC

• Processor cores (Rocket, BOOM, …)
• Memory system (w/ coherence protocol)
• Interconnect (TileLink)
• Custom blocks (e.g. communication, imaging)
• Standard peripheral devices
  • JTAG
  • SPI
  • I2C
  • BootROM
  • …
Chipyard: SoC Generator
There is a Lot of Open-Source Stuff!

- Many open source components:

  - Chisel
  - BOOM Core
  - Diplomacy
  - FireSim
  - FIRRTL
  - Rocket Core
  - TileLink
  - Configuration System
  - RISC-V
  - Accelerators
  - Caches
  - Peripherals
  - HAMMER
  - BAG

**Goal:** Make it easy for small teams to design, integrate, simulate, validate workload performance and tape-out a custom SoC.
Chipyard

Tooling
- Chisel
- FIRRTL
- RISC-V
- BAG

Rocket Chip
- Generators
  - Rocket Core
  - BOOM Core
  - Accelerators
  - TileLink
  - Caches
  - Peripherals

BAG Modules

Flows
- Diplomacy
- Configuration System
- FireSim
- HAMMER
- Software RTL Simulation
What is Rocket Chip?

• A highly parameterizable SoC generator
  • Replace default Rocket core w/ your own core
  • Add your own coprocessor
  • Add your own SoC IP to uncore

• A library of reusable SoC components
  • Memory protocol converters
  • Arbiters and Crossbar generators
  • Clock-crossings and asynchronous queues

• The largest open-source Chisel codebase
  • Scala allows advanced generator features

• Developed at Berkeley, now maintained by many
  • SiFive, CHIPS Alliance, UC Berkeley

In industry: **SiFive Freedom E310**
Structure of a Rocket Chip SoC

Tiles: unit of replication for a core
- CPU (Rocket, BOOM, Ariane)
- L1 Caches
- Page-table walker

L2 banks:
- Receive memory requests

FrontBus:
- Connects to DMA devices

ControlBus:
- Connects to core-complex devices

PeripheryBus:
- Connects to other devices

SystemBus:
- Ties everything together
BOOM: The Berkeley Out-of-Order Machine

- Superscalar RISC-V OoO core
- Fully integrated in Rocket Chip ecosystem
- Open-source
- Described in Chisel
- Parameterizable generator
- Taped-out (BOOM; VLSI’18)
  - Updated: https://boom-core.org/
- Full RV64GC ISA support
  - FP, RVC, Atomics, PMPs, VM, Breakpoints, RoCC
  - Runs real OS’s, software
- Drop-in replacement for Rocket
Saturn-V Vector Core

• Implements the RV vectorextension 1.0
  • Support generating implementations with precise-traps/virtual-memory
  • Full coverage for indexed/strided/segmented memory instructions
  • Full coverage of integer/fixed-point/float-point

• Extensible
  • Support easily adding new instructions, specialized for {DSP/ML/sparsity/etc.}
  • Support extending to new data-types

• Parameterized generator
  • Support combinations of VLEN+DLEN
  • Support different memory interfaces (L1 vs bus access)
  • Generate implementations across multiple design points
  • Should be easy to schedule kernels
RoCC Accelerators

- **RoCC**: Rocket Chip Coprocessor
- Execute custom RISC-V instructions for a custom extension
- Examples of RoCC accelerators
  - Vector accelerators
  - Memcpy accelerator
  - Machine-learning accelerators (Gemmini, NVDLA)
    - See the second part of the tutorial!
  - Java GC accelerator
Gemmini: A Systolic Generator

- Systolic Array Accelerator
- Fully configurable
  - Dataflow – Output/Weight Stationary
  - Dimensions
  - Bitwidths/Datatypes
  - Pipeline Depth
  - Memory capacity
  - Memory banking
  - Memory Bus Width

https://www.github.com/ucb-bar/gemmini
L2 Cache and Memory System

- Multi-bank shared L2
  - SiFive’s open-source IP
  - Fully coherent
  - Configurable size, associativity
  - Supports atomics, prefetch hints

- Non-caching L2 Broadcast Hub
  - Coherence w/o caching
  - Bufferless design

- Multi-channel memory system
  - Conversion to AXI4 for compatible DRAM controllers
Core Complex Devices

- **BootROM**
  - First-stage bootloader
  - DeviceTree

- **PLIC**

- **CLINT**
  - Software interrupts
  - Timer interrupts

- **Debug Unit**
  - DMI
  - JTAG
Other Chipyard Blocks

• **Hardfloat**: Parameterized Chisel generators for hardware floating-point units

• **IceNet**: Custom NIC for FireSim simulations

• **SiFive-Blocks**: Open-sourced Chisel peripherals
  • GPIO, SPI, UART, etc.

• **TestchipIP**: Berkeley utilities for chip testing/bringup
  • Tethered serial interface
  • Simulated block device

• **SHA3**: Educational SHA3 RoCC accelerator
Building Heterogeneous Systems

• Constellation: Open-source NoC generator

Example generated NoCs

https://github.com/ucb-bar/constellation

J. Zhao, NoCArc’22
Supporting Many Accelerators

• ReRoCC (Remote RoCC), a virtualized and disaggregated accelerator integration interface for many-accelerator integration

  J. Zhao, OSCAR’23

• AuRORA, a full-stack methodology for integrating accelerators in a scalable manner for multi-tenant execution

  S. Kim, MICRO’23
Customization

- **Cores and controllers:** Intra-core Rocket/BOOM configurations
  - Control core / PMU as an example
- **Simple RoCC accelerators**
  - SHA3 as an ‘instructional’ demo
- **Complex RoCC accelerators**
  - Hwacha and Gemmini as examples
- **MMIO Tilelink accelerators**
- **Peripherals**
Rocket Chip Configuration

class MyCustomConfig extends Config(
    new WithExtMemSize((1<<30) * 2L) ++
    new WithBlockDevice ++
    new WithGPIO ++
    new WithBootROM ++
    new hwacha.DefaultHwachaConfig ++
    new WithInclusiveCache(capacityKB=1024) ++
    new boom.common.WithLargeBooms ++
    new boom.system.WithNBoomCores(3) ++
    new WithNormalBoomRocketTop ++
    new rocketchip.system.BaseConfig)

TestHarness

Tile 0

Tile 1

Tile 2

SimBlockDevice

SimAXIMem

SysBus

MemBus

GPIOs

BootROM

L1I$

L1D$

Hwacha

L2

L1I$

L1D$

Hwacha

EECS251B L02 CHIPYARD

EECS251B L02 CHIPYARD

Nikolić, Spring 2024
Rocket Chip Configuration

class MyCustomConfig extends Config(
    new WithExtMemSize((1<<30) * 2L) ++
    new WithBlockDevice ++
    new WithGPIO ++
    new WithBootROM ++
    new hwacha.DefaultHwachaConfig ++
    new WithInclusiveCache(capacityKB=1024) ++
    new boom.common.WithLargeBooms ++
    new boom.system.WithNBoomCores(2) ++
    new rocketchip.subsystem.WithNBigCores(1) ++
    new WithNormalBoomRocketTop ++
    new rocketchip.system.BaseConfig)

class MyCustomConfig extends Config{
  new WithExtMemSize((1<<30) * 2L) ++
  new WithBlockDevice ++
  new WithGPIO ++
  new WithBootROM ++
  new WithMultiRoCCGemmini(2) ++
  new WithMultiRoCCSha3(1) ++
  new WithMultiRoCCHwacha(0) ++
  new WithInclusiveCache(capacityKB=1024) ++
  new boom.common.WithLargeBooms ++
  new boom.system.WithNBoomCores(2) ++
  new rocketchip.subsystem.WithNBigCores(1)++
  new WithNormalBoomRocketTop ++
  new rocketchip.system.BaseConfig)

Rocket Chip Configuration

class MyCustomConfig extends Config(
    new WithExtMemSize((1<<30) * 2L) ++
    new WithBlockDevice ++
    new WithGPIO ++
    new WithBootROM ++
    new WithMultiRoCCGemmini(2) ++
    new WithMultiRoCCSha3(1) ++
    new WithMultiRoCCHwacha(0) ++
    new WithInclusiveCache(capacityKB=1024) ++
    new boom.common.WithLargeBooms ++
    new boom.system.WithNBoomCores(2) ++
    new rocketchip.subsystem.WithRV32 ++
    new rocketchip.subsystem.WithNBigCores(1)++
    new WithNormalBoomRocketTop ++
    new rocketchip.system.BaseConfig)

TestHarness

Top

Tile 0

SysBus

MemBus

GPIOs

BootROM

Tile 1

3-w BOOM

Hwacha

L1I$

L1D$

Tile 2

RV32Rocket

Gemmini

L2

Tile 1

SIMAXI Mem

L1I$

L1D$

SimBlockDevice

BooM

L1I$

L1D$
class MyCustomConfig extends Config{
    new WithExtMemSize((1<<30) * 2L) ++
    new WithBlockDevice ++
    new WithGPIO ++
    new WithJtagDTM ++
    new WithBootROM ++
    new WithMultiRoCCGemmini(2) ++
    new WithMultiRoCCSha3(1) ++
    new WithMultiRoCCHwacha(0) ++
    new WithInclusiveCache(capacityKB=1024) ++
    new boom.common.WithLargeBooms ++
    new boom.system.WithNBoomCores(2) ++
    new rocketchip.subsystem.WithRV32 ++
    new rocketchip.subsystem.WithNBigCores(1)++
    new WithNormalBoomRocketTop ++
    new rocketchip.system.BaseConfig
}

TestHarness

Tile 0
- 3-w BOOM
- L1I$
- L1D$
- Hwacha

Tile 1
- 3-w BOOM
- L1I$
- L1D$
- SHA3

Tile 2
- RV32Rocket
- Gemmini
- L1I$
- L1D$

SimBlockDevice

SimAXIMem
Rocket Chip Configuration

class MyCustomConfig extends Config{
  new WithExtMemSize((1<<30) * 2L) ++
  new WithBlockDevice ++
  new WithGPIO ++
  new WithJtagDTM ++
  new WithBootROM ++
  new WithRationalBoomTiles ++
  new WithRationalRocketTiles ++
  new WithMultiRoCCGemmini(2) ++
  new WithMultiRoCCHwacha(0) ++
  new WithInclusiveCache(capacityKB=1024) ++
  new boom.common.WithLargeBooms ++
  new boom.system.WithNBoomCores(2) ++
  new rocketchip.subsystem.WithRV32 ++
  new rocketchip.subsystem.WithNBigCores(1) ++
  new WithNormalBoomRocketTop ++
  new rocketchip.system.BaseConfig)

TestHarness

Top

Tile 0

L1I$
L1D$
3-w BOOM
Hwacha

Tile 1

L1I$
L1D$
3-w BOOM
SHA3

Tile 2

L1I$
L1D$
RV32Rocket
Gemmini

SimBlockDevice
SimAXIMem

SysBus
MemBus
GPIOs
BootROM
JTAG

clk_1

clk_2

clk_0

EECS251B L02 CHIPYARD
Nikolić, Spring 2024
FireSim

- Cycle-exactly simulating large SoCs on cloud FPGAs @10s-100s of MHz
- Open-source: https://fires.im
- Targets:
  1. Architecture evaluation
  2. Validate application on a pre-Si SoC

S. Karandikar, ISCA ’18, IEEE Micro TopPicks ’18, CARRV ’19

Example of (2): PageRank on Rocket+Hwacha

EECS251B L02 CHIPYARD
A. Amid, CARRV’19

SoC RTL Other RTL Network Topology SW Models Full Workload

Automatically deployed, high-performance, distributed simulation

A. Amid, CARRV’19
Hammer

- Modular VLSI flow
  - Allow reusability
  - Allow for multiple “small” experts instead of a single “super” expert
  - Build abstractions/APIs on top
  - Improve portability
  - Improve hierarchical partitioning

- Three categories of flow input
  - Design-specific
  - Tool/Vendor-specific
  - Technology-specific

Customized TCL Script
Simulation/Implementation Targets

• Custom hardware design is not just about generated IP blocks!
• Different collaterals for different simulation or implementation targets
  • Design cycle RTL simulation
  • Verification / validation
  • VLSI flow
Software

• Compatible standard RISC-V Tools versions
• ESP-Tools as a non-standard equivalent SW tools package with custom accelerator extensions (Hwacha, Gemmini)
• Improved BareMetal testing flow
  • Use libgloss and newlib instead of in-house syscalls
• FireMarshal workload management
Summary

• We will use Chipyard to generate a minimalist RISC-V SoC for logic design and circuits experiments

• Labs will exercise the design flow

• Think of projects that can:
  • Test a circuit idea in a larger system
  • Design a block to improve SoC
    • Co-processor/Accelerator
    • Peripheral device
Next Lecture

- Chisel