EE290C - Fall 2018

Advanced Topics in Circuit Design

VLSI Signal Processing

TuTh 2 - 3:30pm
Cory 540A/B

Practical Information

- **Instructor:**
  - **Borivoje Nikolić**
    - 509 Cory Hall , 3-9297, bora@eecs
    - Office hours: Th 10-11am or by appointment
- **GSI: Paul Rigge**
  - rigge@berkeley

Class Discussion
http://piazza.com/class#spring2018/ee290c/
Sign up for Piazza!

Class Web page
http://bcourses.berkeley.edu

Class Github
https://github.com/ucberkeley-ee290c

2

Class Objectives

- Review of signal processing algorithms used in wireless and wireline communications, data storage
- Fixed-function and programmable DSP architectures
- Chisel as a hardware-construction language
- Design, verification and validation of a DSP block
- Integration of DSP functions into a complex SoC
- Build an exciting SoC!

3

Prerequisites

- EECS 251A – Introduction to Digital Systems
  - Primarily logic design; FPGA and ASIC tool knowledge is helpful
- EE 123 Digital Signal Processing
  - Need to know FIR/IIR filters, design
  - DFT, FFT, Z transform
  - Graduate DSP knowledge will be reviewed in class

4

Class Materials

- **No perfect textbook**
  - Chueh, Tsai, Lai, Baseband Receiver Design for Wireless MIMO-OFDM Communications, Wiley 2012.
  - Woods, McAllister, Lightbody, Yi, FPGAImplementation of DSP Systems, Wiley 2017.
  - Meher, Stouraitis, Arithmetic Circuits for DSP Applications, Wiley 2017.
  - Markovic, Brodersen, DSP Architecture Design Essentials, Springer, 2012.
- **References**
- **Weekly reading assignments**

5

Rough Class Outline

1. Introduction: systems and optimization metrics (this class)
2. Dataflow analysis
3. DSP arithmetic (Number representations, quantization effects, functions, CORDIC)
4. Automatic gain control using LMS and variants
5. Filters, FIR, IIR. Transformations, pipelining, parallelism, interleaving, transposing distributed arithmetic.
6. Adaptive filters implementation
7. FFT, Frequency-domain filtering
8. Viterbi algorithm and implementation, Soft decoding & turbo.
9. Message passing with LDPC example
10. Systolic arrays, Machine learning accelerators
11. MIMO algorithms
12. Carrier/timing recovery, estimation, filters, correlators.
13. ADC specs, analog impairments
14. SW/HW tradeoffs, modem example
15. Class presentations

6

Class Tracks (August-September)

	Lectures	Assignments	Reading
1	Intro	Chisel bootcamp	Chisel papers
2	Dataflow, numbers	Chisel bootcamp	
3	CORDIC	Chisel bootcamp, CORDIC	CORDIC
4	Using Rocket Chip DSP in Chisel	OFDM simulator	OFDM
5	LMS, AGC	Diplomacy	LMS chapter
6	FIR, IIR filters	FIR design	Filters
7	FFT	FFT design	FFT papers

7

Grading

- **Assignments – 30%**
- **Project – 70%**
  - Report and presentations at the end of semester

8

Tools

- **Chisel**
  - Open and free
- **Logic simulator**
  - Verilator, Synopsys VCS, Cadence Xcelium
- **Logic synthesis**
  - Synopsys DC + 28nm SAED library or
  - Xilinx Vivado
  - Place and route is optional

9

EE290C - Fall 2018

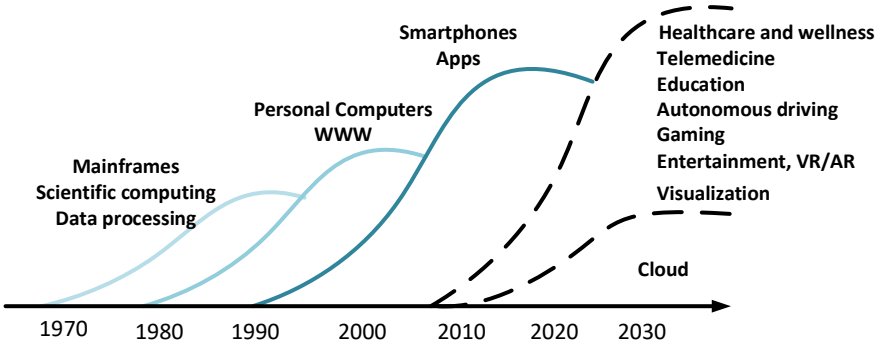
Advanced Topics in Circuit Design

VLSI Signal Processing

DSP systems

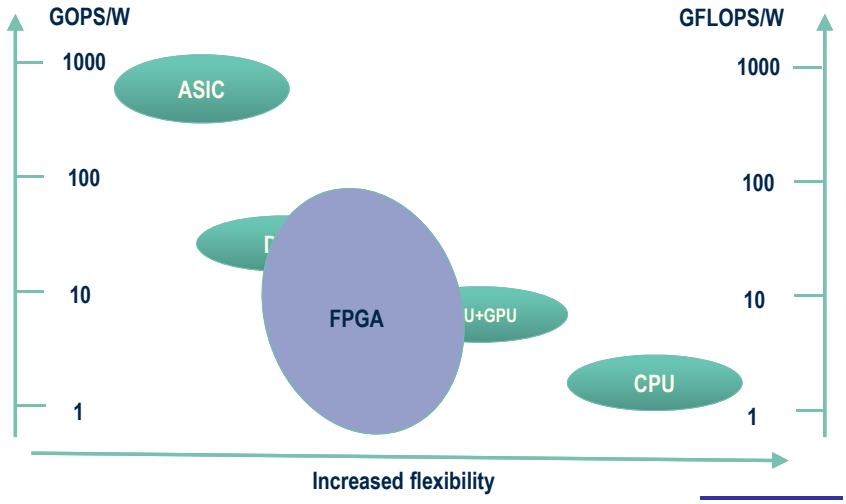
The Big Picture

- Driving applications are diversifying



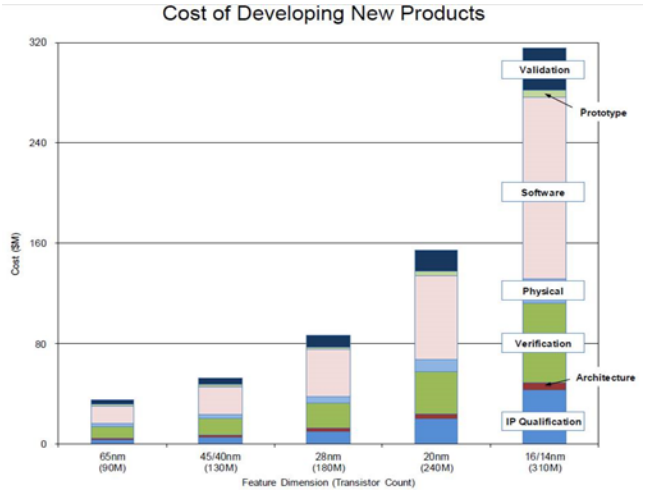
11

Specialization for Energy Efficiency



12

ASICs are Expensive



[Source: IBS]

- **Cost doubling in each generation**

13

Why Wireless?

- **> 1 Billion cell phones an year – mostly smartphones**
  - Major industry driver
- **Wireless embedded into everything from TVs to gaming to clothes**
  - Internet of things yet to come
- **100's of GOPS to TOPS of signal processing done in a tiny form factor with energy constraints**
- **RF/Analog/DSP/Protocols need to be jointly designed and optimized**
- **So – a great driver for the class**
- **(wireline is good too, many shared blocks)**

14

Chisel: Constructing Hardware In a Scala Embedded Language

- Hardware Construction Language (HCL)
- Started in 2010 (DAC 2012)
- HCL based in Scala, a functional/object-oriented programming language
- Hosts Berkeley hardware projects
- Many ongoing industry collaborators (Intel, LBNL, Northrop Grumman, Google, SiFive, BAE Systems)

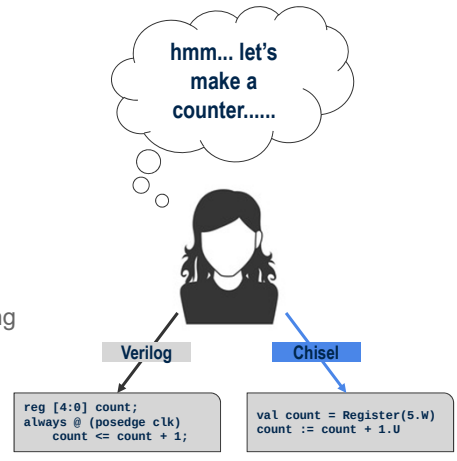
```
import chisel3._

class GCD extends Module {
  val io = IO(new Bundle {
    val a  = Input(UInt(32.W))
    val b  = Input(UInt(32.W))
    val e  = Input(Bool())
    val z  = Output(UInt(32.W))
    val v  = Output(Bool())
  })
  val x = Reg(UInt(32.W))
  val y = Reg(UInt(32.W))
  when (x > y)    { x := x -% y }
  .otherwise      { y := y -% x }
  when (io.e) { x := io.a; y := io.b }
  io.z := x
  io.v := y === 0.U
}
```

15

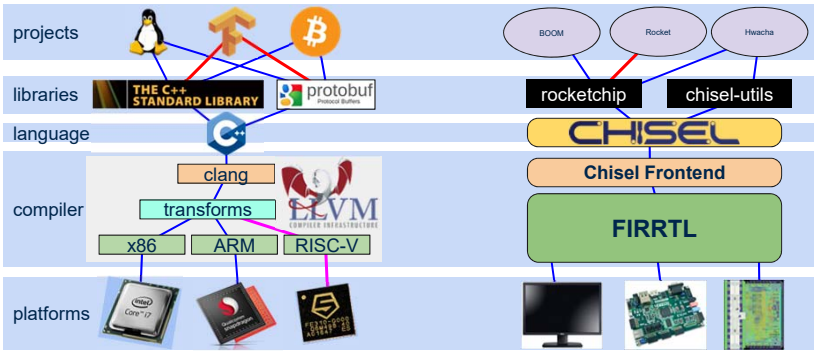
Chisel is RTL, not HLS

- HCLs give users access to the expressiveness of the host language
- Scala provides many features:
  - Powerful parameterization
  - Functional programming
  - Static typing
  - Object-oriented programming
- Enables Agile Design



16

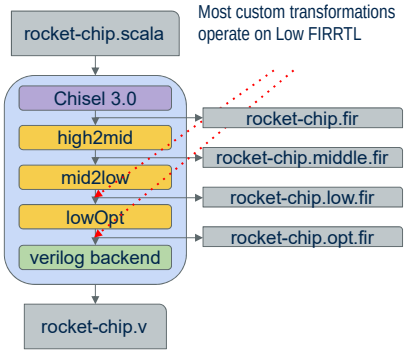
Borrow Good Ideas From Software, Build a Compiler!



17

FIRRTL: Flexible Intermediate Representation for RTL

- A frontend parses and translates source into an intermediate representation, FIRRTL
- Apply lowering transformations that rewrite the design using a simpler and smaller subset of FIRRTL nodes
- Custom transformations make project- or platform-specific changes without modifying the RTL!
- Custom transformations are composable!
- A backend emits design in *.v
- A. Izraelevitz, ICCAD 2017



18

First Assignment

- **Read**
  - J. Bachrach, DAC'12 paper
  - A. Izraelevitz, ICCAD'17 paper
  - A. Wang, DAC'18 paper
- **Chisel bootcamp**
  - Complete modules 1, 2.1, 2.2 by next Thursday
  - We will go through the Chisel bootcamp in 3 weeks

19

So, How to Build an SoC in Class?

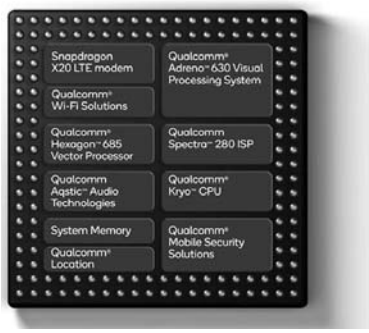
- **Principles (algorithms, architectures) in class**
  - Reinforce through assignments
  - Practice writing Chisel generators through assignments
- **Build Chisel generators**
- **Plug into a template RISC-V SoC**

- **This is really complex, but many details are abstracted**

20

What is a Wireless SoC Today?

- **Qualcomm Snapdragon 850**



- **Complex SoC:**
  - LTE, WiFi modems
  - GPS
  - Applications processors
  - GPU
  - Vector processors
  - Image signal processor
  - Audio signal processor
  - Security
  - System memory

21

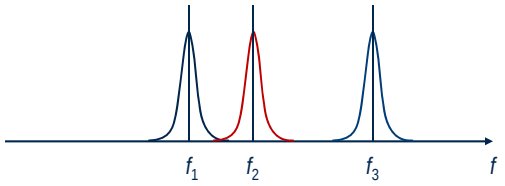
Wireless Modems

- **There are several systems and many standards**
  - Cellular (4G/5G)
  - WiFi
  - Bluetooth
  - Zigbee
  - NB-IoT, LTE-M
  - NFC
- **Similarities and differences**
- **Many shared functions**

22

Frequency Division Multiplexing

- Different users simultaneously use different frequencies

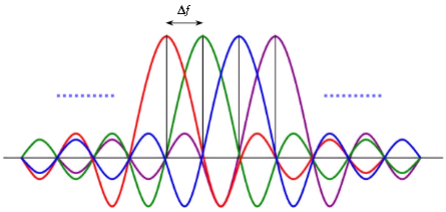


- Inter-carrier interference limited by filtering

23

OFDM Basics

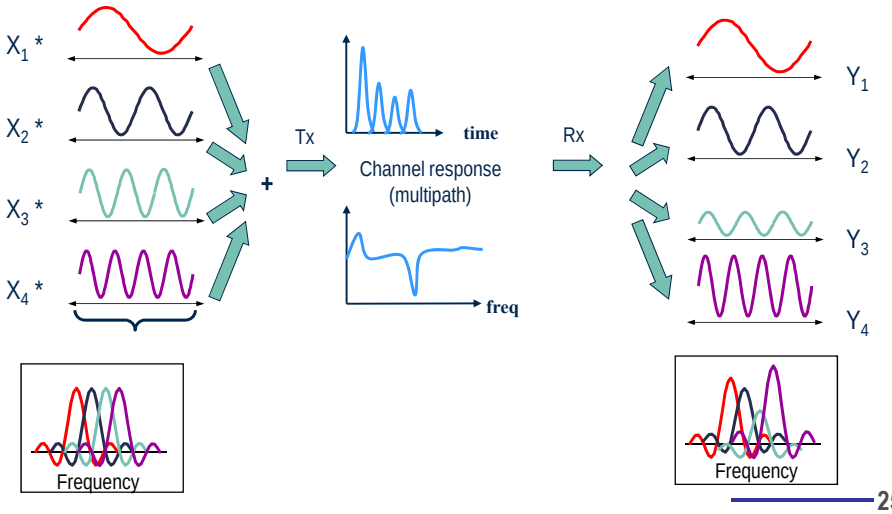
- Orthogonal Frequency Division Modulation (OFDM)
- Multi-carrier system, with fixed inter-carrier spacing



- N Sub-carriers can be efficiently generated via N-point IFFT

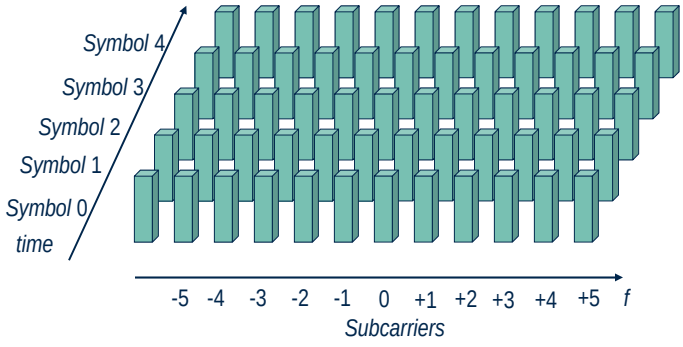
24

OFDM Basics (cont.)



25

OFDM Basics (cont.)

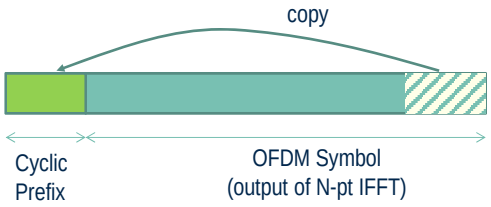


- Each subcarrier can have different modulation
- Preamble
- Pilots

26

Cyclic Prefix

- Cyclic pre-fix added to combat inter-symbol interference due to multi-path and relax synchronization
- Also enables frequency domain equalization since channel appears as a circular convolution
- Sub-carriers remain orthogonal as long as multi-path delay spread is smaller than CP length



27

OFDMA

- MA = Multiple Access
- Different users allocated different groups of subcarriers

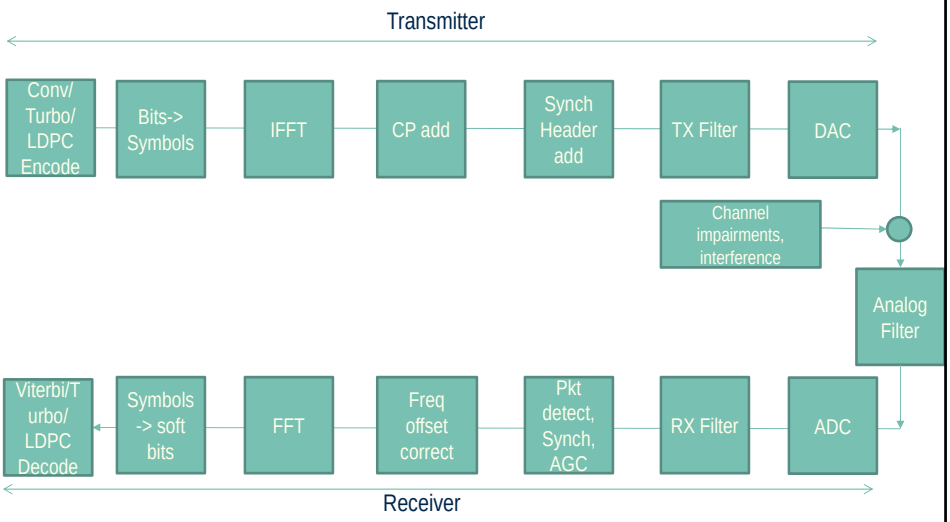
28

OFDM Use

- **WPAN**
  - Bluetooth 3.0, 4.0, 5.0
- **WLAN**
  - IEEE 802.11a/g, 802.11n, 802.11ac, 802.11ad,…
- **Cellular**
  - LTE, 5G
- **Broadcast**
  - DVB-T, DVB-H, DAB

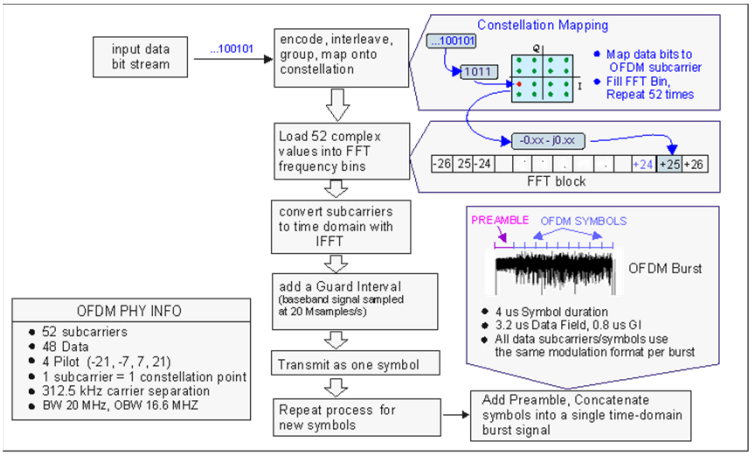
29

Generic OFDM Modem



30

OFDM Signal Generation



802.11a OFDM Signal Generation Process

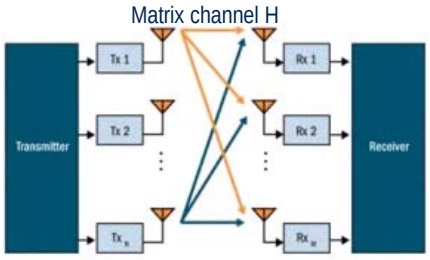
- From Keysight

http://rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/subsystems/wlan-ofdm/content/ofdm_basicprinciplesoverview.htm

31

Multiple-Input-Multiple-Output (MIMO)

- Use of multiple receive/transmit antennas to increase spectral efficiency

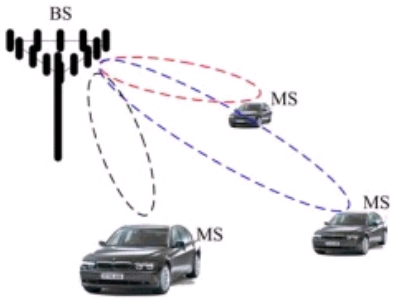


- As long as channel H is well conditioned it can be effectively inverted and the spatial streams are recovered
- MIMO support is part of cellular data and WLAN standards

32

Multi-user MIMO

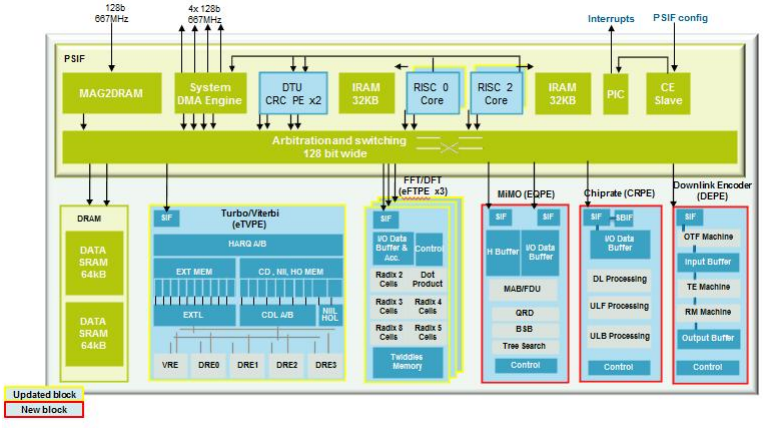
- Beamforming to each user
- Spatial multiplexing to increase spectral efficiency
- Complex matrix operations performed in real time



33

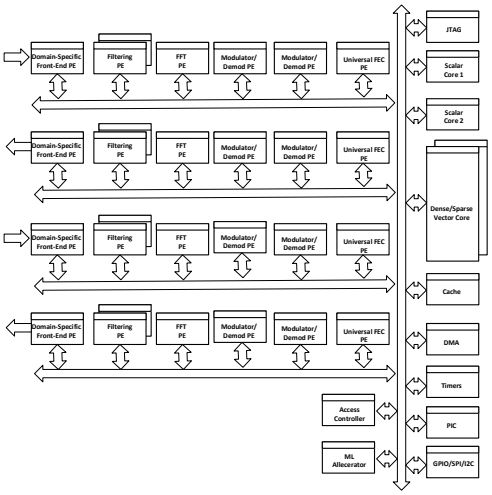
Software-Defined Basebands

- Freescale LTE accelerator (~2010)



34

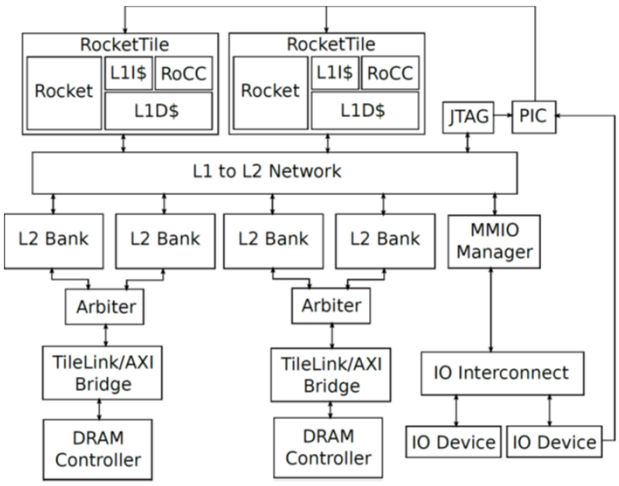
Rough SDR Platform for This Class



- Signal processing blocks added to a RISC-V Rocket

35

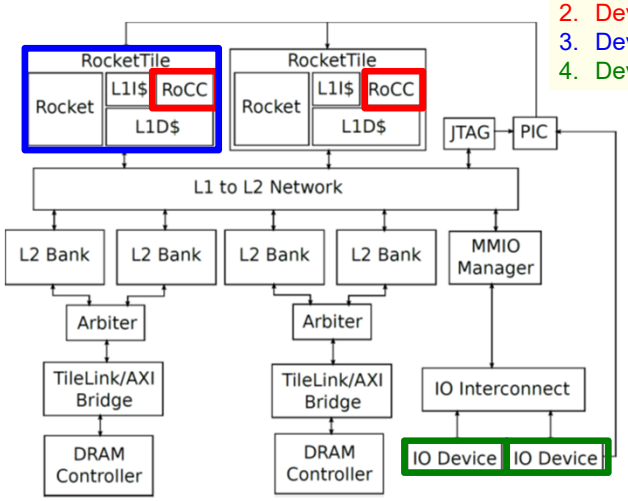
RISC-V "Rocket Chip" SoC Generator Example Output



36

"Rocket Chip" SoC Specialization

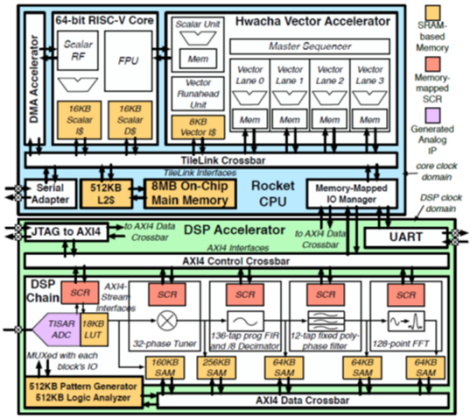
1. Change Parameters
2. Develop New Accelerators
3. Develop Own RISC-V Core
4. Develop Own Device



37

Example SoC Based on Rocket

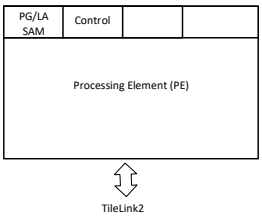
- Signal analysis SoC
  https://github.com/ucb-art/craft2-chip



38

Attaching Peripherals

- **DSP functions attached to Rocket chip via Diplomacy**
  - We will provide examples (and assignments)



39

Next Lecture

- **Number representations**
- **Methods of computation**

40