Designing RISC-V *from the .pdf* / A baseband processor extension to the ISA

Cecil Accetti R. de A. Melo – cecaccetti@sjtu.edu.cn

6th RISC-V workshop – Shanghai, China
May 8-10 2017

北斗导航与位置服务重点实验室
Shanghai Key Laboratory of Navigation and Location-based Services

Agenda

Introductions

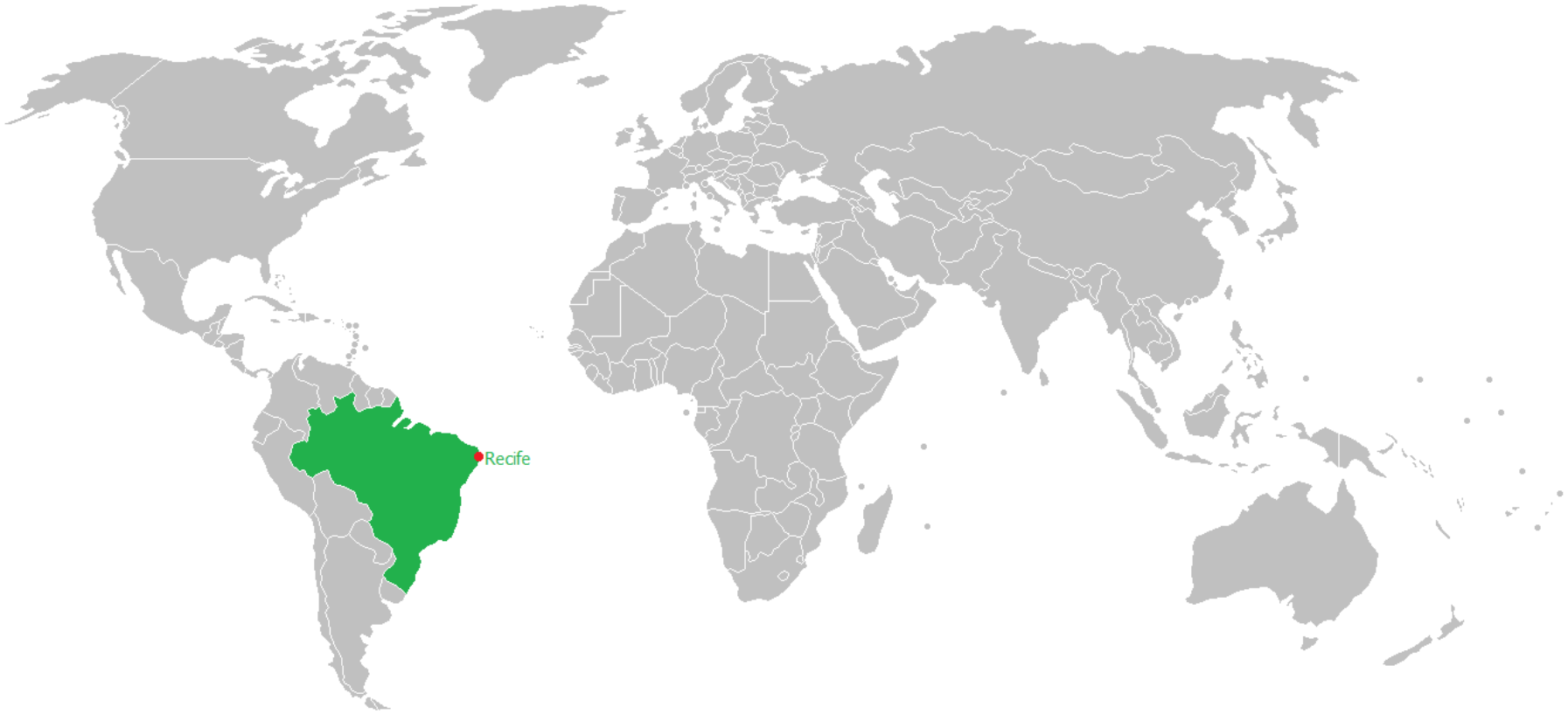
Embedded (SoC) system design for 3rd parties

Some reasons to develop your own CPU core

Designing RISC-V from the pdf

Oolong : A baseband processor extension for RISC-V

Next steps @ SJTU


Recife

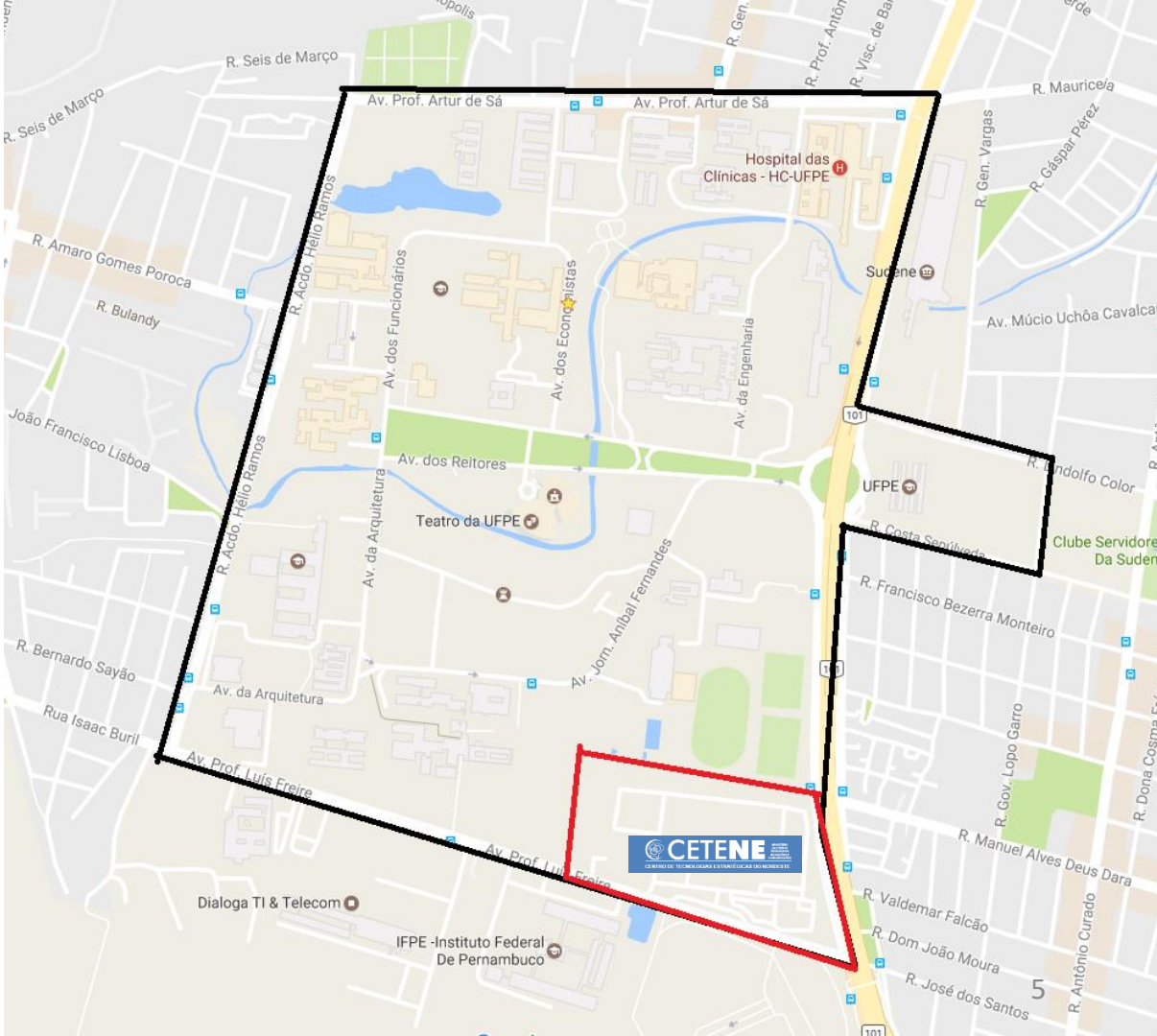
Origins

UNIVERSIDADE FEDERAL DE PERNAMBUCO

VIRTUS IMPAVIDA

Centro de Informática UFPE

CETENE
MINISTÉRIO DA CIÊNCIA TECNOLOGIA INOVAÇÕES E COMUNICAÇÕES
CENTRO DE TECNOLOGIAS ESTRATÉGICAS DO NORDESTE



Embedded (SoC) system design for 3rd parties

Usual problem:

Fit block A with blocks B and C for some task

Embedded (SoC) system design for 3rd parties

Usual problem:

Fit block A with blocks B and C for some task, but under US$ x,

Embedded (SoC) system design for 3rd parties

Usual problem:

Fit block A with blocks B and C for some task, but under US$ x and it should be easy to use and maintain.

Embedded (SoC) system design for 3rd parties

Usual problem:

Fit block A with blocks B and C for some task, but under US$ x and it should be easy to use and maintain.

We could design all IP cores, but we'll still need a CPU to glue it all

Design your own CPU core

Why?

#1 Economics

Development HW != Release HW

Design your own CPU core

Why?

#1 Economics

Development HW != Release HW

Stratix III Development Kit

USD 5,495

Design your own CPU core

Why?

#1 Economics

Development HW != Release HW

Stratix III Development Kit
USD 5,495

Cyclone III FPGA 22KLEs
~ USD 60
Board cost under USD 150

Design your own CPU core

Why?

#2 Owning the IP

We needed a CPU, but which one? Which ISA? How about the tools?

Our options:

NIOS : Licensing and Royalties $

ARM : Licensing, royalties (and other costs) $$$

OpenRISC

Homebrew 8051 core

Design your own CPU core

Why?

#2 Owning the IP

We needed a CPU, but which one? Which ISA? How about the tools?

Our options:

NIOS : Licensing and Royalties $

ARM : Licensing, royalties (and other costs) $$$

OpenRISC

Homebrew 8051 core

*Instruction sets **do** matter*

Design your own CPU core

Why?

#3: ***"What I cannot create, I do not understand"***

Richard Feynman (1918-1988)

Opportunity to design something that I'd want to use

http://blogs.scientificamerican.com/oscillator/files/2013/07/feynman.jpg

EE Times Connecting the Global Electronics Community

Home | News | Opinion | Messages | Authors | Video | Slideshows | Teardown | Education | EELife

designlines | Android | Automotive | Embedded | Industrial Control | Internet of Things | MCU | Medical | Memo

BREAKING NEWS NEWS & ANALYSIS: Cadence DSP Grabs Neural Nets

designlines SoC

Blog

RISC-V: An Open Standard for SoCs

The case for an open ISA

Krste Asanović & David Patterson, UC Berkeley
8/7/2014 07:55 AM EDT
65 comments post a comment

3 saves
LOGIN TO RATE

Tweet Share 138 G+1 20

Just as Linux has become the standard OS for most computing devices, Berkeley researchers envision RISC-V becoming the standard ISA for all computing devices.

http://www.eetimes.com/author.asp?doc_id=1323406

Design your own CPU core

RISC-V

- 131 page (readable) specification (User-level 2.1)
- Stable tools (GCC)
- “Make your own implementation”
- Open source ISA
- Efficient ISA
- Open, active community

Design your own CPU core

RISC-V

- 131 page (readable) specification (User-level 2.1)
- Stable tools (GCC)
- “Make your own implementation”
- Open source ISA
- Efficient ISA
- Open, active community

Designing RISC-V from the pdf

Pros

Complete freedom for architecture choice

Incremental development – Most needed instructions first

Fit the target device (FPGA)

The RISC-V Instruction Set Manual
Volume I: User-Level ISA
Version 2.1

Andrew Waterman, Yunsup Lee, David Patterson, Krste Asanović
CS Division, EECS Department, University of California, Berkeley
{waterman|yunsup|pattrsn|krste}@eecs.berkeley.edu
May 31, 2016

This document is also available as Technical Report UCB/EECS-2016-118.

Designing RISC-V from the pdf

Cons

The CPU is just part of the problem.

We need a **Bus, Interrupts, Peripherals**, OS, and software (!) to make things work.

No standard way of interfacing with all those components.

Infrastructure is on its way, but not there yet

The RISC-V Instruction Set Manual
Volume I: User-Level ISA
Version 2.1

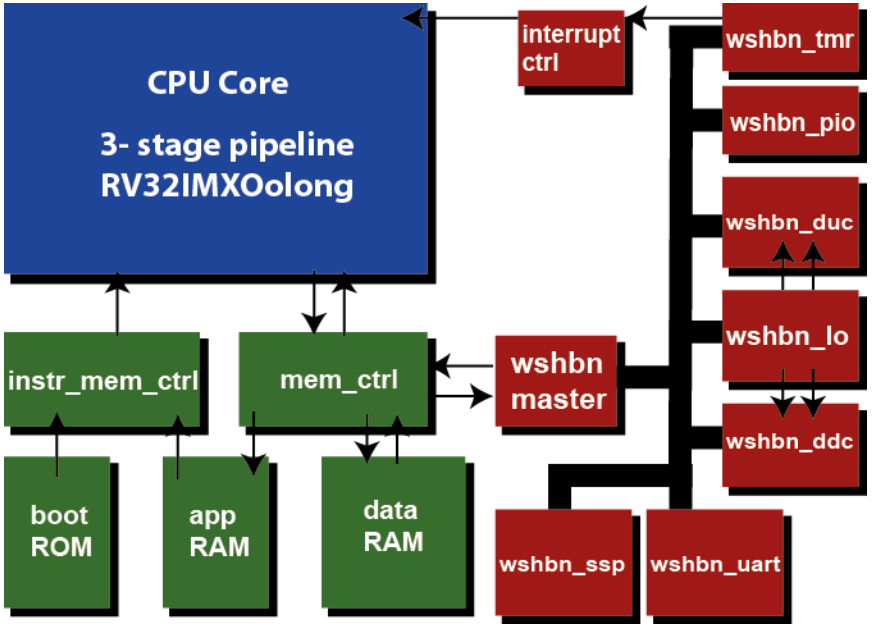
Andrew Waterman, Yunsup Lee, David Patterson, Krste Asanović
CS Division, EECS Department, University of California, Berkeley
{waterman|yunsup|pattrsn|krste}@eecs.berkeley.edu
May 31, 2016

This document is also available as Technical Report UCB/EECS-2016-118.

Oolong （乌龙）RV32IMXOolong

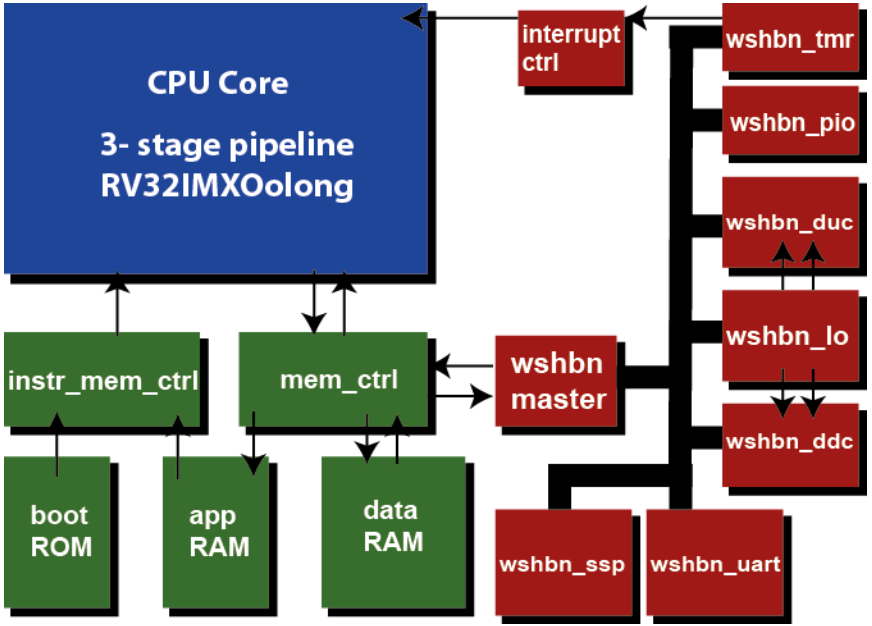
Microcontroller applications

- RV32IM ISA
- ~5000LEs (Altera)
- I/O
  - GPIO, UART, SPI,I²C, DACs and ADCs (external)
- Simple Access to peripherals
  - WISHBONE Bus, 2 to 3 clock cycles wait
  - Simple Bus -> Simple driver
    - More space in ROM for actual programs



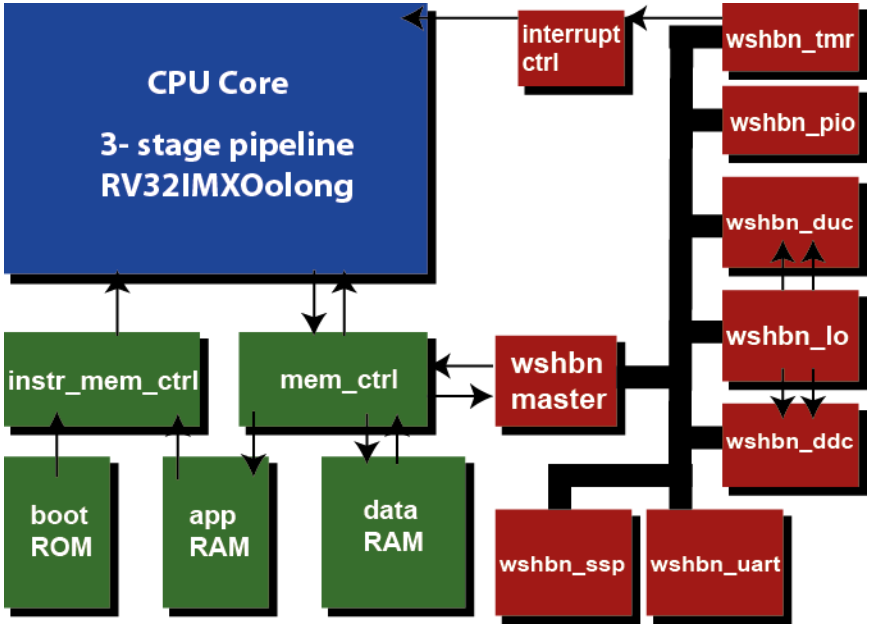
Oolong （乌龙）RV32IMXOolong

- 3-stage pipeline
- M-mode only
- Static Branch prediction
- Harvard architecture
  - Boot ROM, Application RAM, Data RAM
- Interrupts
  - 2 Timers
  - Peripherals
- Up to 1.36 DMIPS/MHz



Oolong （乌龙）RV32IMXOolong

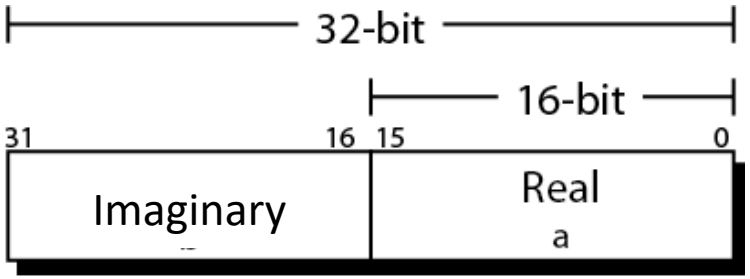
- Uses FPGA multipliers (1 cycle latency)
- Instruction Set Extension for SDR
  - Software-defined Radio
  - Complex number arithmetic (fixed point)
    - Quadrature Modulation
    - Complex Multiply-accumulate
  - Galois Field GF(2) multiplication
    - AES256: 32% faster decryption
  - 8 bit SIMD
    - Voice, data, characters



Oolong （乌龙）RV32IMXOolong

Extended ISA

- 32 bit word (16 bit imaginary, 16 bit real)
  - ADCs and DACs usually work with <16bit words
  - 60dB SFDR for Quadrature Modulation



$A = a + jb$

0000000	rs2	rs1	000	rd	0001011	CADD rd, rs1, rs2
0100000	rs2	rs1	000	rd	0001011	CSUB rd, rs1, rs2
0010000	rs2	rs1	000	rd	0001011	CCONJ rd, rs1, rs2
0010000	rs2	rs1	001	rd	0001011	CMUL rd, rs1, rs2
0000110	rs2	rs1	001	rd	0001011	CABS rd, rs1, rs2
0000100	rs2	rs1	001	rd	0001011	CSMUL rd, rs1, rs2
0000010	rs2	rs1	001	rd	0001011	CMAC rd, rs1, rs2
0000000	rs2	rs1	011	rd	0001011	PXMUL rd, rs1, rs2
0000010	rs2	rs1	011	rd	0001011	VMUL rd, rs1, rs2
0100000	rs2	rs1	011	rd	0001011	VMAC rd, rs1, rs2
0000001	rs2	rs1	011	rd	0001011	VMUL rd, rs1, rs2
0000100	rs2	rs1	011	rd	0001011	VMAC rd, rs1, rs2

Oolong （乌龙）RV32IMXOolong

Extended ISA

- Complex Multiplication
  - (a + jb) * (c + jd) = (ac – bd) + j(ad + bc) : 8 instructions (cycles)

```
int cmplx_mac_re (int a_re,int a,int b, int c, int d)
{
return(a_re +( a*c - b*d)) ;

}


int cmplx_mac_im (int a_im,int a,int b, int c, int d)
{
return(a_im +( a*d + b*c)) ;

}
```

a)

```
<cmplx_mac_re>:
 mul     a1,a1,a3
 mul     a4,a2,a4
 sub     a1,a1,a4
 add     a0,a1,a0
 ret
<cmplx_mac_im>:
 mul     a1,a1,a4
 mul     a3,a2,a3
 add     a1,a1,a3
 add     a0,a1,a0
 ret
```

b)

```
cmac a0, a1,a2
```

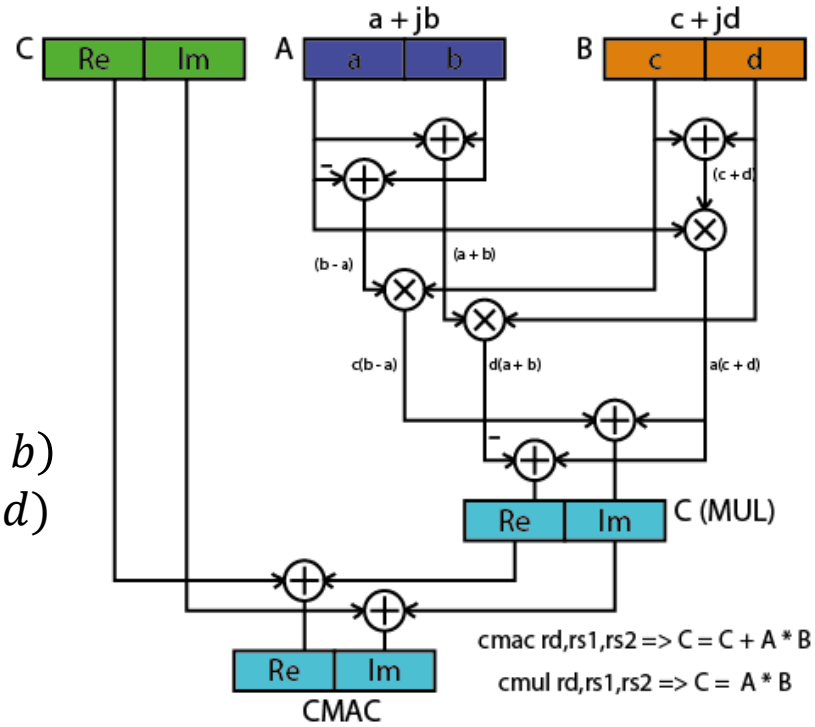
c)

Oolong （乌龙）RV32IMXOolong

Extended ISA

- Complex Multiplication
  - (a + jb) * (c + jd) = (ac – bd) + j(ad + bc) : 8 instructions
  - CMAC, CMUL
  - 20% faster MODEM

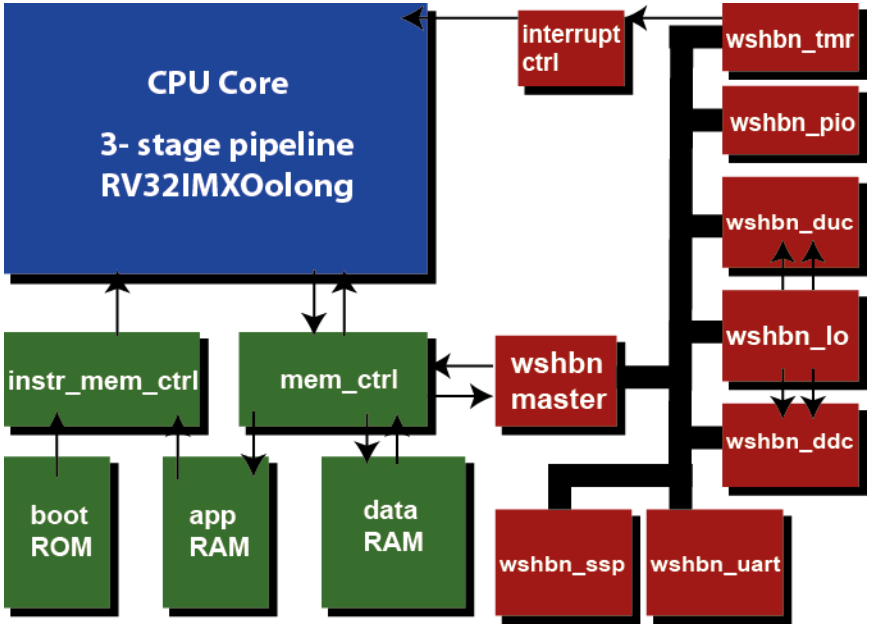
$$Re\{(a+jb)\times(c+jd)\} = a(c+d) - d(a+b)$$
$$Im\{(a+jb)\times(c+jd)\} = c(b-a) + a(c+d)$$



Oolong （乌龙）RV32IMXOolong

Peripherals

- UART
- Interrupt Controller
- GPIO
- SPI/I²C
- Numerically controlled Oscillator (NCO)
- Digital Upconverter
- Digital Downconverter
- And more…



Next steps: RISC-V development @SJTU

Accelerator Design

Neural networks

Computer Vision

SLAM

Programming Languages

Functional

Non-Von Neumann

Instruction v. Accelerators

RISC-V CPU as the hub