A Design of RISC-V RV32IMAFC Core for MCU

as an Opensource IP

mmRISC-1

Nov. 19, 2021

Munetomo Maruyama (圓山宗智)

@Processing_Unit

Rev.01

Munetomo Maruyama (圓山宗智)

As a Hobbyist for 43 years
Design of Hardwares and Softwares

Activity talking Today

As an Engineer for 35 years
Development of Micro Controllers and SoCs

As a Freelance Technical Writer for 20 years
Technical Magazines and Books

RISC-V Impact

Specification

Design

Development Works

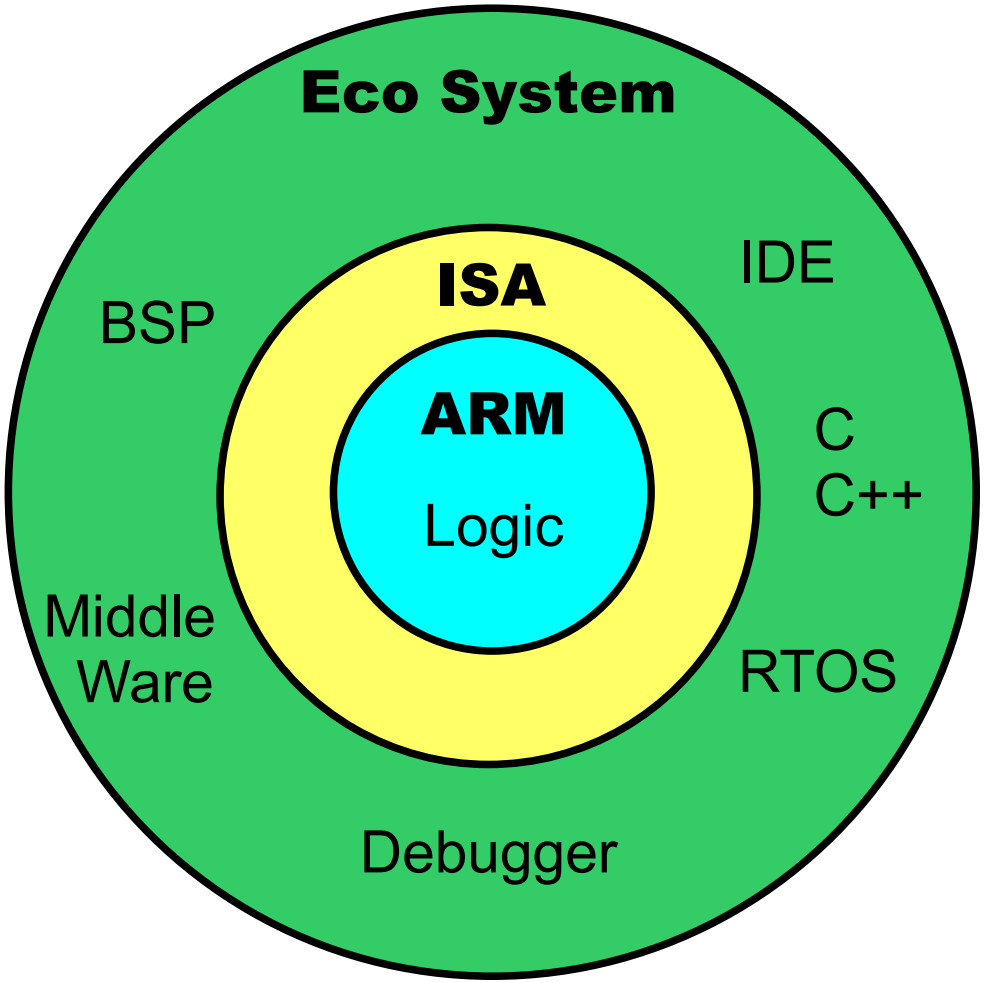
Summary

RISC-V Impact

Popular CPU Core : ARM

Pros : Comprehensive Eco System

Cons: License Fee / Royalty Fee



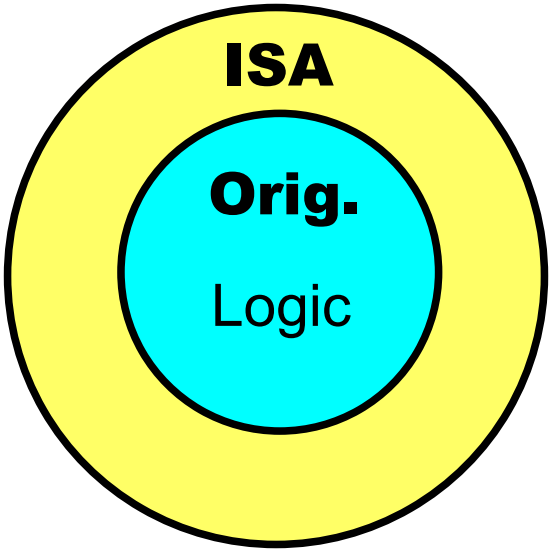
Self-Made Original Core

Pros : No License / Royalty Fee

Fun! Learn a Lot!

Cons: Poor Eco System

Not Practical



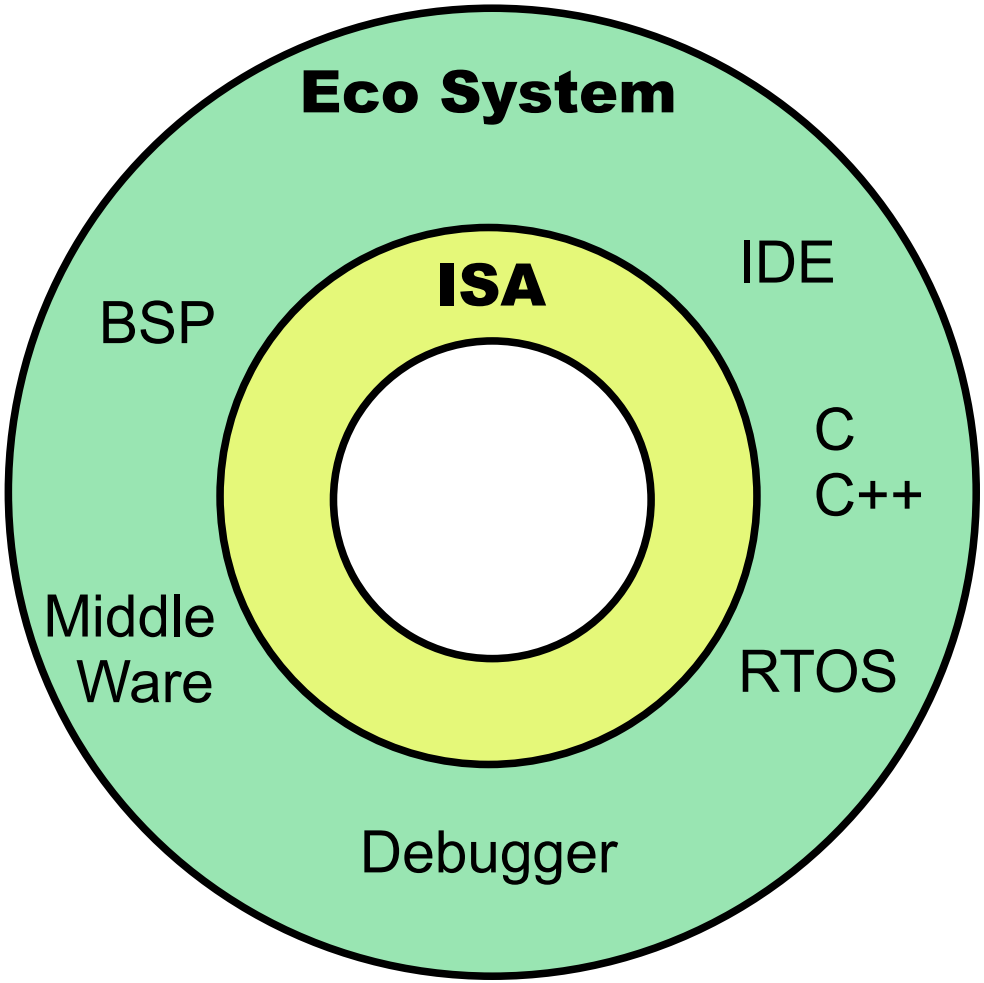
Assembler…only?

RISC-V Essential

Pros : Comprehensive Eco System

Cons: No Hardware Logics

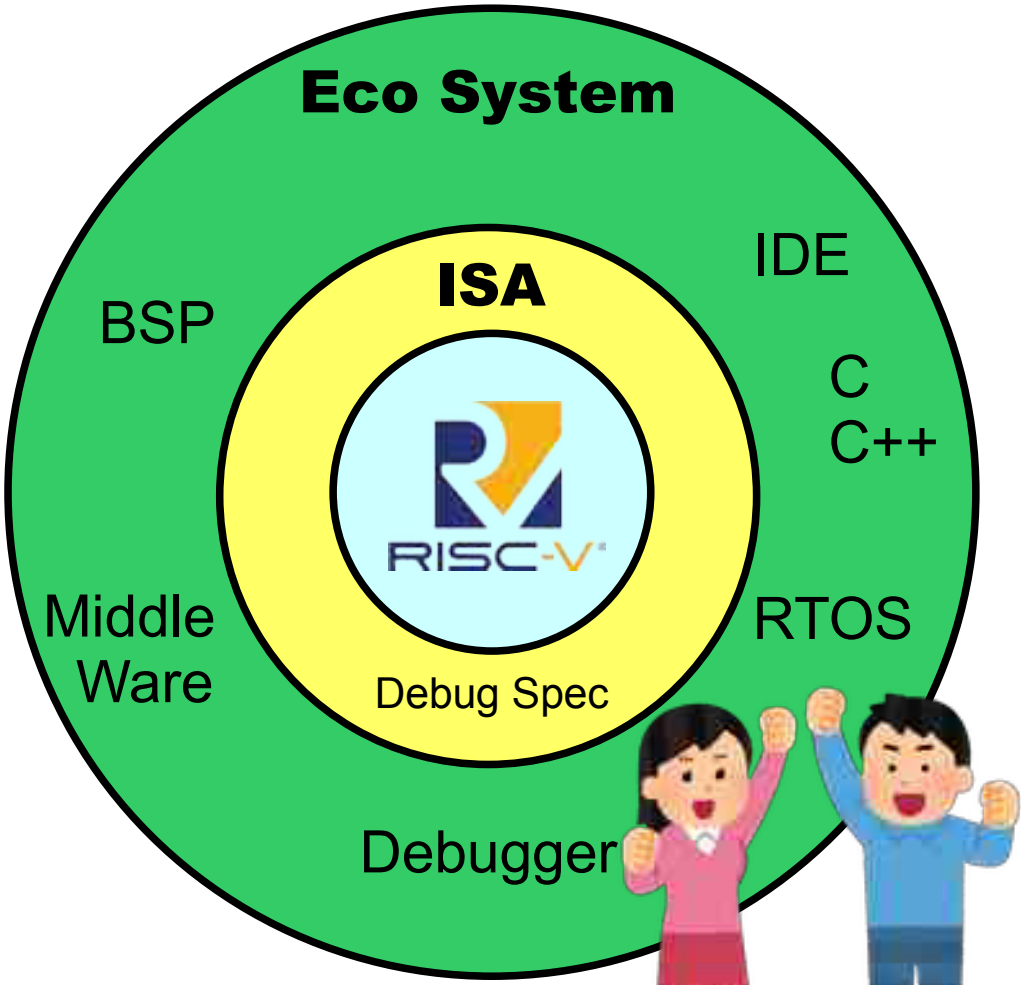
Big Impact !



Self-Made RISC-V Core

Pros : Comprehensive Eco System

only No License / Royalty Fee



Why I have made by myself?

As Public Stance

- To have practical CPU core
- To get large eco-system world
- To have white-box CPU
- To contribute to MCU designers
- To learn more and get technical experience more
- To educate freshers by giving it as sample design
- To provide technical articles, technical books, technical seminars

As Private Stance (real motif)

- *Anyway, To enjoy !*

Specification

Target Product and Application

ASSP MCU for Industrial / Consumer Applications

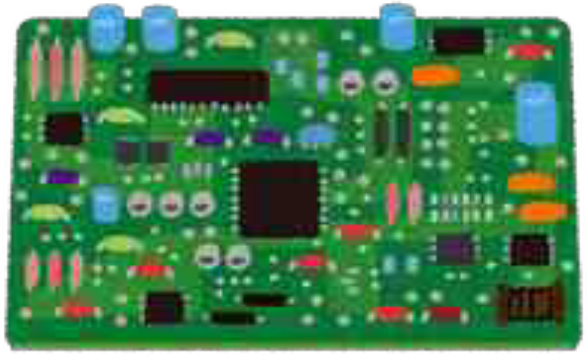
Feedback Control Systems

Motor Control : FOC, Servo

Power Converter : AC-DC, DC-DC, DC-AC

Robot

Drone



Control Board

ISA Determination

RV32IMAFC

Fast Digital Filter Operations

32bits Fixed Point, 1cyc Multiplication **(M)**

32bit Floating Point, 1cyc FADD/FMUL/FMADD **(F)**

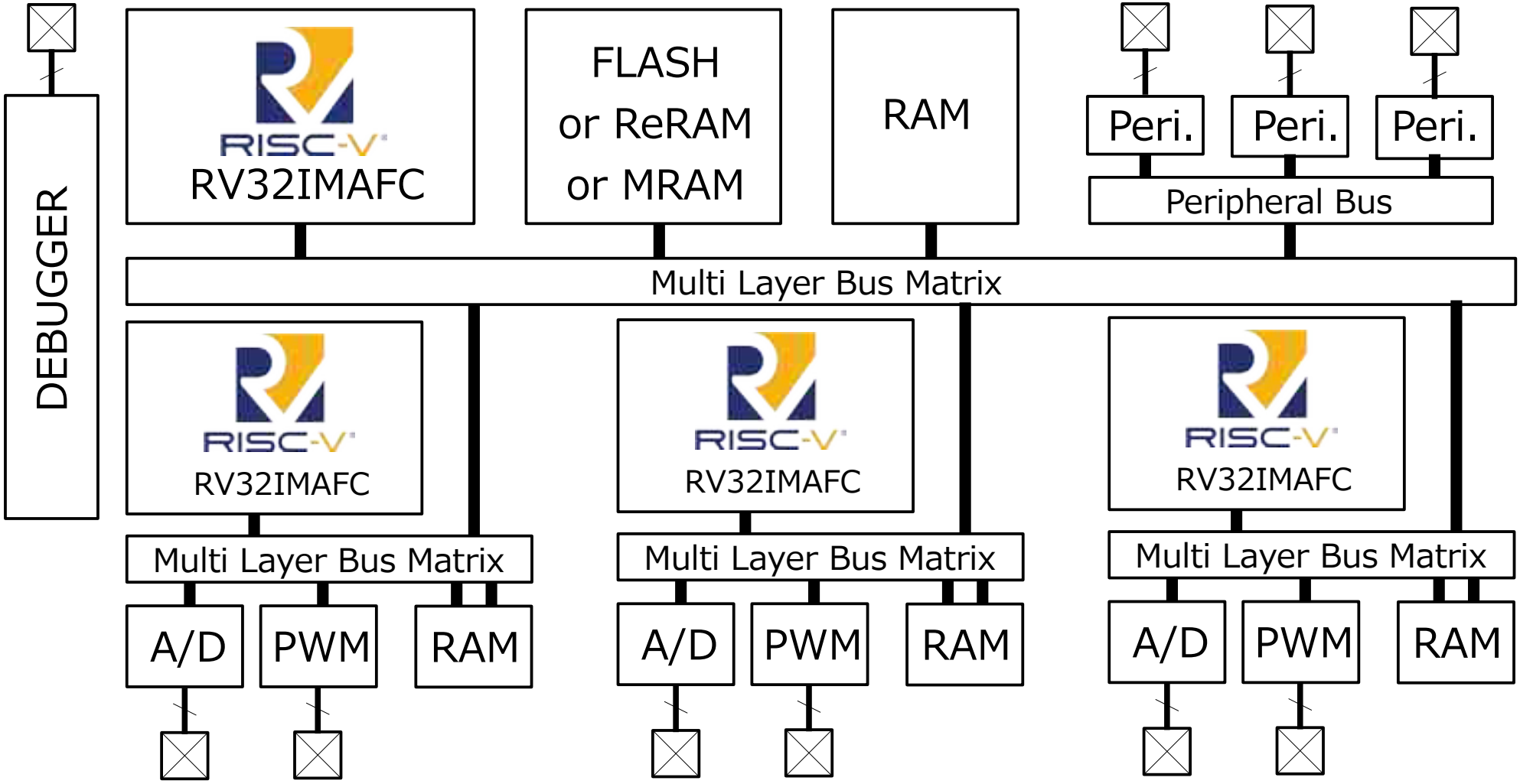
Small Embedded Memory Size

Compressed Code Size **(C)**

MCU → Low Operating Frequency

No Task Switching by Interrupt

Multi-Cores / Heterogeneous / Domain Specific **(A)**



Specification

Item	Description
Core Name	mmRISC-1 (Much More RISC)
ISA	RV32IM[A][F]C
Supported Hart Counts	1 hart to 1,000,000 harts
Privileged Mode	Machine Mode only
Pipeline	3 to 5 stages for Integer 5 to 6 stages for Floating Point
32bit Integer Multiplication	1-cycle
32bit Floating Point	1-cycle (FADD.S/FMUL.S/FMADD.S)
Debug Support	External Debug Support Ver.0.13.2 JTAG Interface Hardware Triggers x 4
Interrupts	External + Machine Software + Machine Timer + User IRQ : 64 inputs, independent vectors 16 Priority Levels for each, allow nested
Bus I/F	AHB Lite, Instruction / Data / LR-SC Monitor / Debug Access
RTL	Verilog-2001, System Verilog
System Proof	FPGA + OpenOCD + Eclipse

Debug Support is important for practical use.

Implemented Instructions

RV32I

```
LUI    rd, imm
AUIPC  rd, imm
ADD    rd, rs1, rs2
SUB    rd, rs1, rs2
ADDI   rd, rs1, imm
SLL    rd, rs1, rs2
SRL    rd, rs1, rs2
SRA    rd, rs1, rs2
SLLI   rd, rs1, shamt
SRLI   rd, rs1, shamt
SRAI   rd, rs1, shamt
XOR    rd, rs1, rs2
OR     rd, rs1, rs2
AND    rd, rs1, rs2
XORI   rd, rs1, imm
ORI    rd, rs1, imm
ANDI   rd, rs1, imm
SLT    rd, rs1, rs2
SLTU   rd, rs1, rs2
SLTI   rd, rs1, imm
SLTIU  rd, rs1, imm
BEQ    rs1, rs2, imm
BNE    rs1, rs2, imm
BLT    rs1, rs2, imm
BGE    rs1, rs2, imm
BLTU   rs1, rs2, imm
BGEU   rs1, rs2, imm
JAL    rd, imm
JALR   rd, imm
LB     rd, rs1, imm
LH     rd, rs1, imm
LW     rd, rs1, imm
LBU    rd, rs1, imm
LHU    rd, rs1, imm
SB     rs1, rs2, imm
SH     rs1, rs2, imm
SW     rs1, rs2, imm
FENCE
ECALL
EBREAK
```

RV32 Zifencei

```
FENCE.I
```

RV32 Zicsr

```
CSRRW  rd csr, rs1
CSRRS  rd csr, rs1
CSRRC  rd csr, rs1
CSRRWI rd, csr, imm
CSRRSI rd, csr, imm
CSRRCI rd, csr, imm
```

RV32M

```
MUL    rd, rs1, rs2
MULH   rd, rs1, rs2
MULHSU rd, rs1, rs2
MULHU  rd, rs1, rs2
DIV    rd, rs1, rs2
DIVU   rd, rs1, rs2
REM    rd, rs1, rs2
REMU   rd, rs1, rs2
```

RV32A

```
LR.W      rd, rs1
SC.W      rd, rs1, rs2
AMOSWAP.W rd, rs1, rs2
AMOADD.W  rd, rs1, rs2
AMOXOR.W  rd, rs1, rs2
AMOAND.W  rd, rs1, rs2
AMOOR.W   rd, rs1, rs2
AMOMIN.W  rd, rs1, rs2
AMOMAX.W  rd, rs1, rs2
AMOMINU.W rd, rs1, rs2
AMOMAXU.W rd, rs1, rs2
```

RV32C

```
C.LI       rd, imm
C.LUI      rd, imm
C.ADD      rd, rs2
C.SUB      rd', rs2'
C.ADDI     rd, imm
C.ADDI4SPN rd', uzuimm
C.ADDI16SP sp, sp, imm
C.SLLI     rd, shamt
C.SRLI     rd', shamt
C.SRAI     rd', shamt
C.XOR      rd', rs2'
C.OR       rd', rs2'
C.AND      rd', rs2'
C.ANDI     rd',  imm
C.MV       rd, rs2
C.BEQZ     rs1', imm
C.BNEZ     rs1', imm
C.J        imm
C.JR       rs1
C.JAL      imm
C.JALR     rs1
C.LW       rd', rs1', imm
C.LWSP     rd, imm
C.SW       rs1', rs2', imm
C.SWSP     rs2, imm
C.EBREAK
C.NOP
ILLEGAL
```

RV32F

```
FADD.S     fd, fs1, fs2
FSUB.S     fd, fs1, fs2
FMUL.S     fd, fs1, fs2
FDIV.S     fd, fs1, fs2
FSQRT.S    fd, fs1
FMADD.S    fd, fs1, fs2,
fs3
FMSUB.S    fd, fs1, fs2,
fs3
FNMADD.S   fd, fs1, fs2,
fs3
FNMSUB.S   fd, fs1, fs2,
fs3
FMIN.S     fd, fs1, fs2
FMAX.S     fd, fs1, fs2
FSGNJ.S    fd, fs1, fs2
FSGNJN.S   fd, fs1, fs2
FSGNJX.S   fd, fs1, fs2
FMV.X.W    rd, fs1
FMV.W.X    fd, rs1
FCVT.W.S   rd, fs1
FCVT.WU.S  rd, fs1
FCVT.S.W   fd, rs1
FCVT.S.WU  fd, rs1
FEQ.S      rd, fs1, fs2
FLT.S      rd, fs1, fs2
FLE.S      rd, fs1, fs2
FCLASS.S   rd, fs1
FLW        fd, rs1, imm
FSW        rs1, fs2, imm
```

RV32FC

```
C.FLW      fd', rs1', imm
C.FLWSP    fd, imm
C.FSW      rs1', fs2', imm
C.FSWSP    fs2, imm
```

Implemented CSRs and other Resources

Machine Mode CSR

MVENDORID
MARCHID
MIMPID
MSTATUS
MISA
MIE
MIP
MTVEC
MSCRATCH
MEPC
MCAUSE
MTVAL
MCYCLE
MCYCLEH
MINSTRET
MINSTRETH
MCOUNTINHIBIT
CYCLE
TIME
INSTRET
CYCLEH
TIMEH
INSTRETH

FPU CSR

FCSR
FRM
FFLAGS
FCONV

Interrupt Controller CSR

MINTCURLVL
MINTPRELVL
MINTCFGENABLE0
MINTCFGENABLE1
MINTCFGSENSE0
MINTCFGSENSE1
MINTPENDING0
MINTPENDING1
MINTCFGPRIORITY0
MINTCFGPRIORITY1
MINTCFGPRIORITY2
MINTCFGPRIORITY3
MINTCFGPRIORITY4
MINTCFGPRIORITY5
MINTCFGPRIORITY6
MINTCFGPRIORITY7

Debug / Trigger CSR

DCSR
DPC
TSELECT
MCONTROL(TDATA1)
ICOUNT(TDATA1)
TDATA2
TINFO

Memory Mapped MTIME

MTIME_CTRL
MTIME_DIV
MTIME
MTIMEH
MTIMECMP
MTIMECMPH
MSOFTIRQ

DTM JTAG Reg.

BYPASS
IDCODE
DTMCS
DMI
BYPASS

DM Reg.

DMSTATUS
DMCONTROL
HARTINFO
HAWINDOWSEL
HAWINDOW
AUTHDATA
HALTSUM0
HALTSUM1
HALTSUM2
HALTSUM3
DATA0
DATA1
COMMAND
COMMAND
ABSTRACTCS
SBCS
SBADDRESS0
SBDATA0

Interrupts and Exceptions

- IRQ_MSOFT / IRQ_MTIME / IRQ_EXT
- IRQ[63:0] : 16 level priorities, Sensing Mode (Level / Edge)
- EBREAK / CBREAK / ECALL / Break Point / Misaligned Access / Bus Error

Cause and Vector						
Group	MCAUSE		Vector Offset		Description	MTVAL
	Interrupt	Exception Code	Direct Mode	Vectored Mode		
Asynchronous	1	0x03	base	base + 0x000c	Machine Software Interrupt (priority is controlled by software)	0x00000000
	1	0x07	base	base + 0x001c	Machine Timer Interrupt (priority is controlled by software)	0x00000000
	1	0x0b	base	base + 0x002c	Machine External Interrupt (priority is controlled by software)	0x00000000
	1	0x10	base	base + 0x0040	Machine IRQ00 (priority is controlled by hardware)	0x00000000
	1	0x11	base	base + 0x0044	Machine IRQ01 (priority is controlled by hardware)	0x00000000
	1	...	...	...	...	...
	1	0x4f	base	base + 0x013c	Machine IRQ63 (priority is controlled by hardware)	0x00000000
Synchronous	0	0x00	base	base	Instruction Address Misaligned	Never occurs
	0	0x01	base	base	Instruction Access Fault	Fault Instruction Address
	0	0x02	base	base	Illegal Instruction	0x00000000
	0	0x03	base	base	Breakpoint (Trigger, EBREAK, CBREAK)	Next PC of executed
	0	0x04	base	base	Load Address Misaligned	Bus Error Address
	0	0x05	base	base	Load Access Fault	Bus Error Address
	0	0x06	base	base	Store/AMO(Atomic Memory Operation) Address Misaligned	Bus Error Address
	0	0x07	base	base	Store/AMO Access Fault	Bus Error Address
	0	0x0b	base	base	Environment Call from M-mode	0x00000000

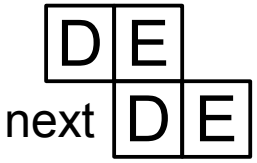
Design

CPU Pipeline

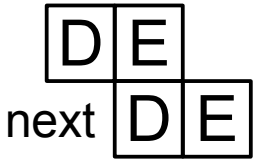
Pipeline Stages

F	Instruction Fetch
D	Decode
E	Execution
M	Data Memory Access
W	Write Back

ALU

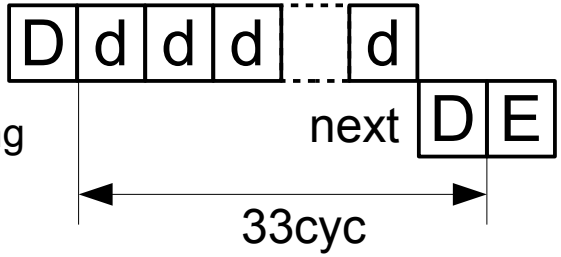


MUL

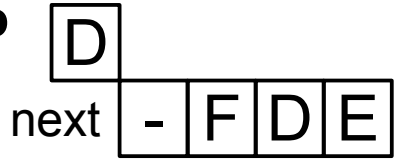


DIV

Non-Restoring

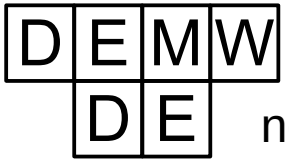


JUMP



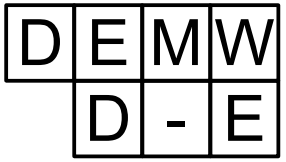
Bcc

LOAD



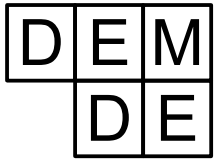
no register conflict

LOAD



register conflict
forwarding W→ E

STORE



FPU Pipeline

FADD

	D	i	a	f
FADD		D	i	a	f

FMUL

	D	i	m	f
FMUL		D	i	m	f

FMADD

	D	i	m	a	f
FMADD		D	i	m	a	f

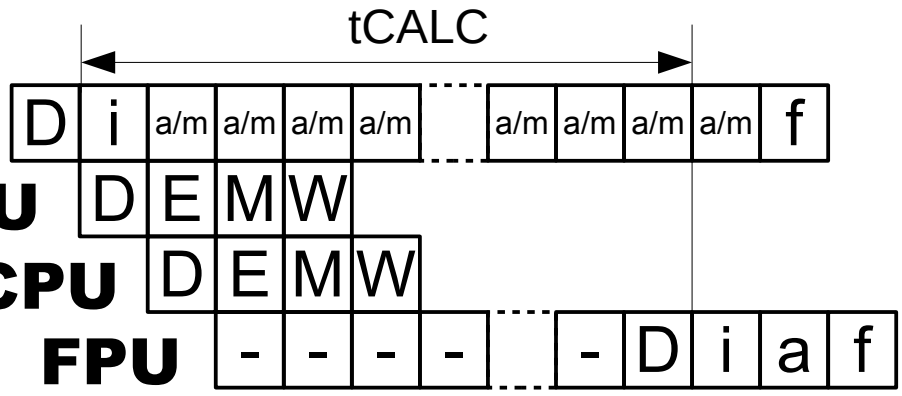
Pipeline Stage

- i : Initial : Convert Float to Internal Format
- a : Add : Add Internal Format
- m : Mul : Multiply Internal Format
- f : Finalize : Convert Internal Format to Float

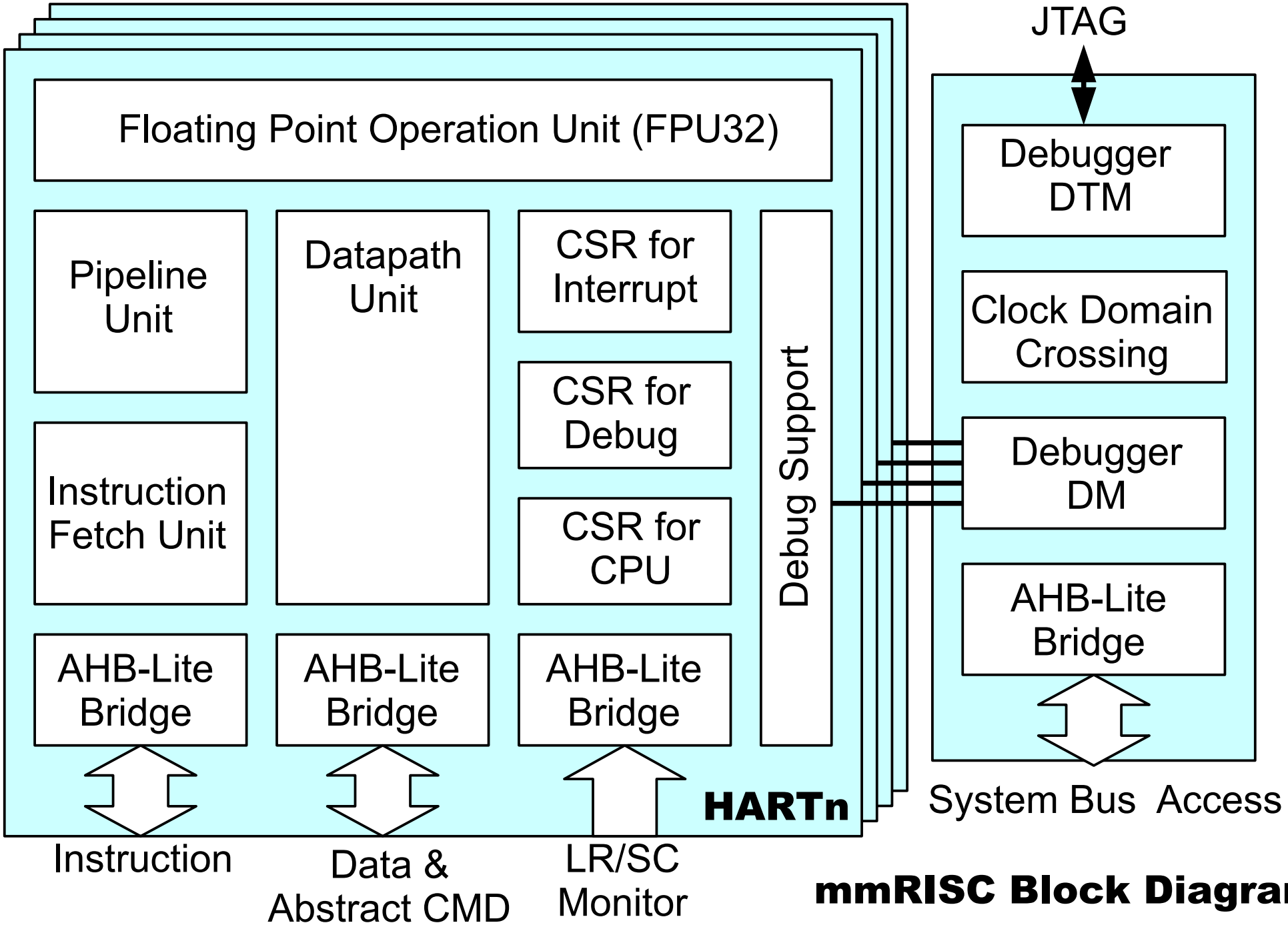
FMADD

	D	i	m	a	f
FADD		-	D	i	a	f

FDIV/FSQRT



Goldschmidt's Algorithm
tCALC=
FDIV: 11cyc (4-loops)
FSQRT: 19cyc (4-loops)



mmRISC Block Diagram

Development Works

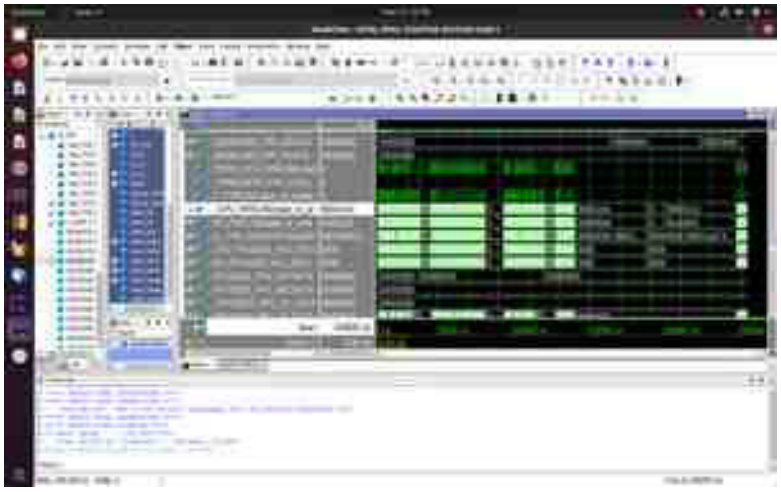
Design Progress

Worked only weekend
Started on Aug. 2017 → Suspended in 2018-2019 → Finished on Sep. 2021

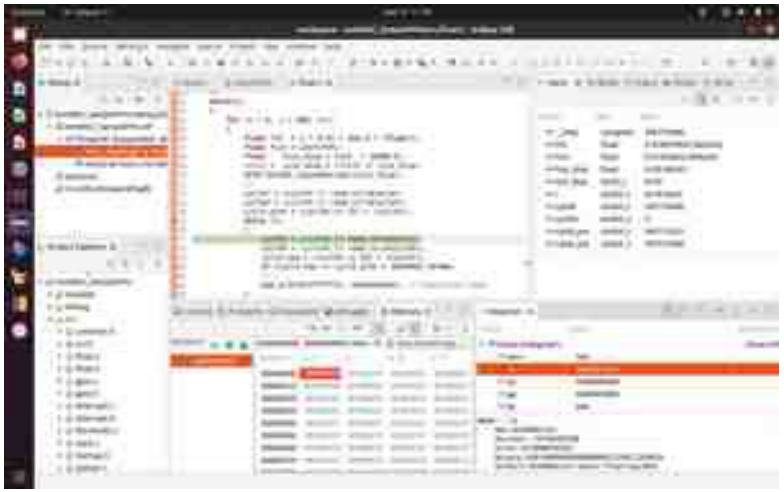
1. Specification and Micro Architecture
2. Test Bench, design from Top Layer
3. **JTAG DTM, Check OpenOCD Communication tentatively**
4. Multi Layer AHB Matrix
5. **Debugger Function except for Triggers, Check Bus Access from JTAG**
6. Instruction Fetch Unit
7. Pipeline Unit / Datapath Unit for basic ALU / Load / Store / Jump Instruction
8. CSR Block and Interrupt Controller
9. Trigger Module in Debugger Function
10. **Functional C Model of Non-Restoring Division, and Convert to RTL**
11. Privileged Instructions
12. Verify whole ISA using “riscv-arch-test”
13. Eclipse, C/C++, OpenOCD Operation and Measure Dhrystone using FPGA
14. **Functional C Model of IEEE754, and Convert to RTL**
15. Floating Point Instructions and CSR
16. Verify whole ISA using ‘riscv-tests”

Platform	Ubuntu 64bit **FREE** (Parallels Desktop on MacBook Pro)
Simulator	Mentor ModelSim Intel FPGA Starter Edition **FREE**
IDE	Eclipse IDE for Embedded C/C++ Developers **FREE**
Tool Chain	github.com/riscv-collab/riscv-gnu-toolchain **FREE**
Open OCD	github.com/riscv/riscv-openocd **FREE**

ModelSim



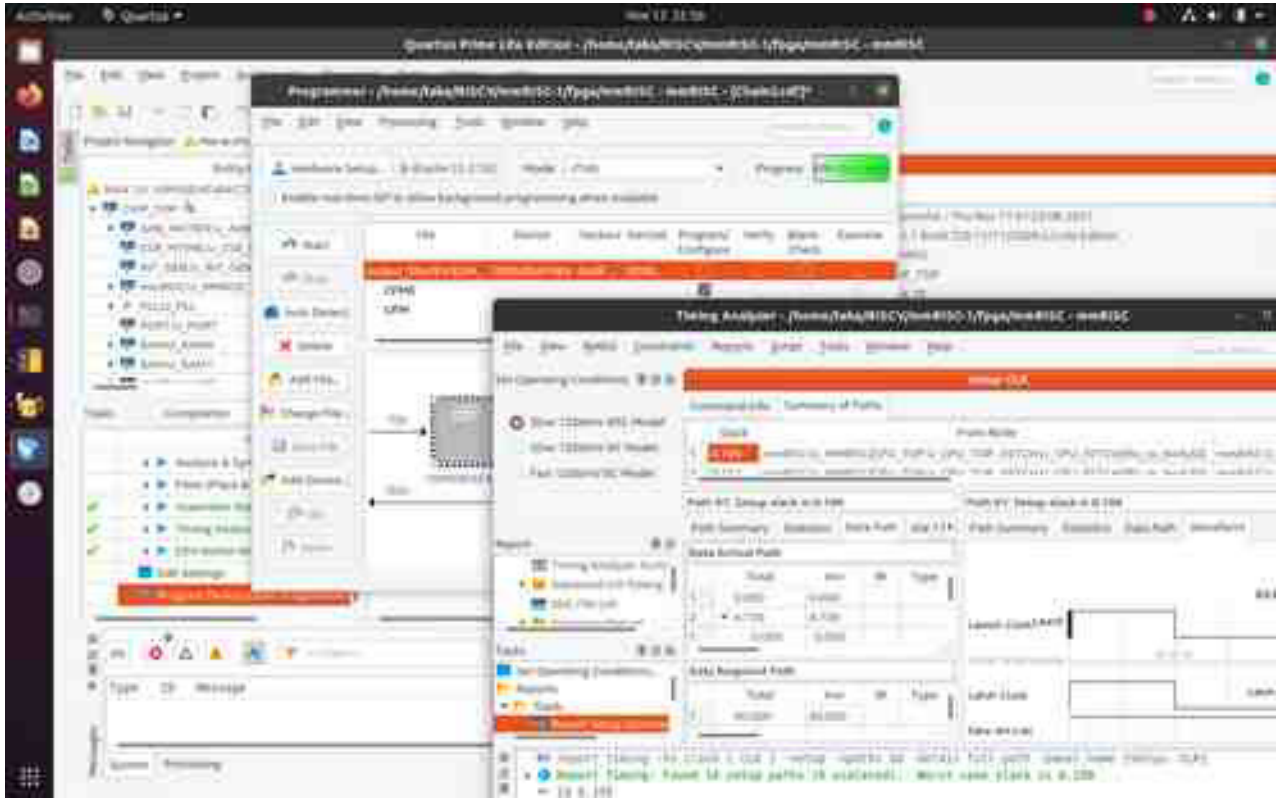
Eclipse



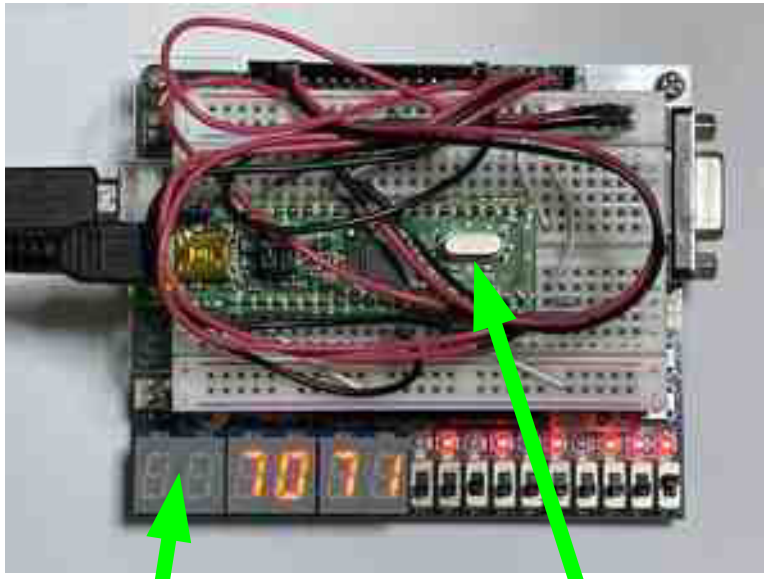
FPGA Board : Terasic DE10-Lite **$95**
(FPGA : Intel MAX 10 10M50DAF484C7G)
Tool : Quartus Prime Lite Edition **FREE**

USB-JTAG OLIMEX ARM-USB-OCD(-H) or compatible **< $20**

Quartus Prime



FPGA Board



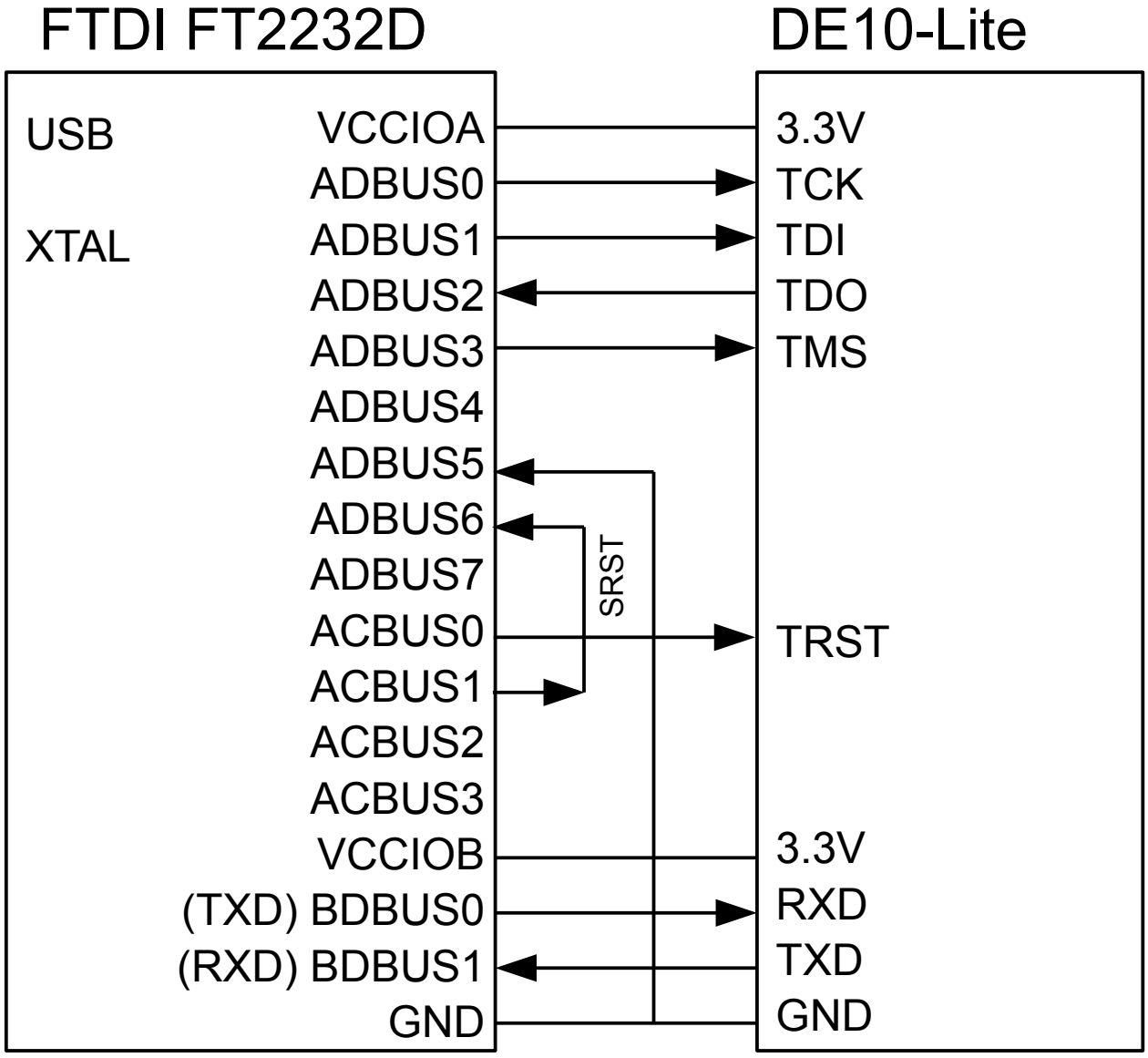
FTDI FT2232D
as USB-JTAG Interface

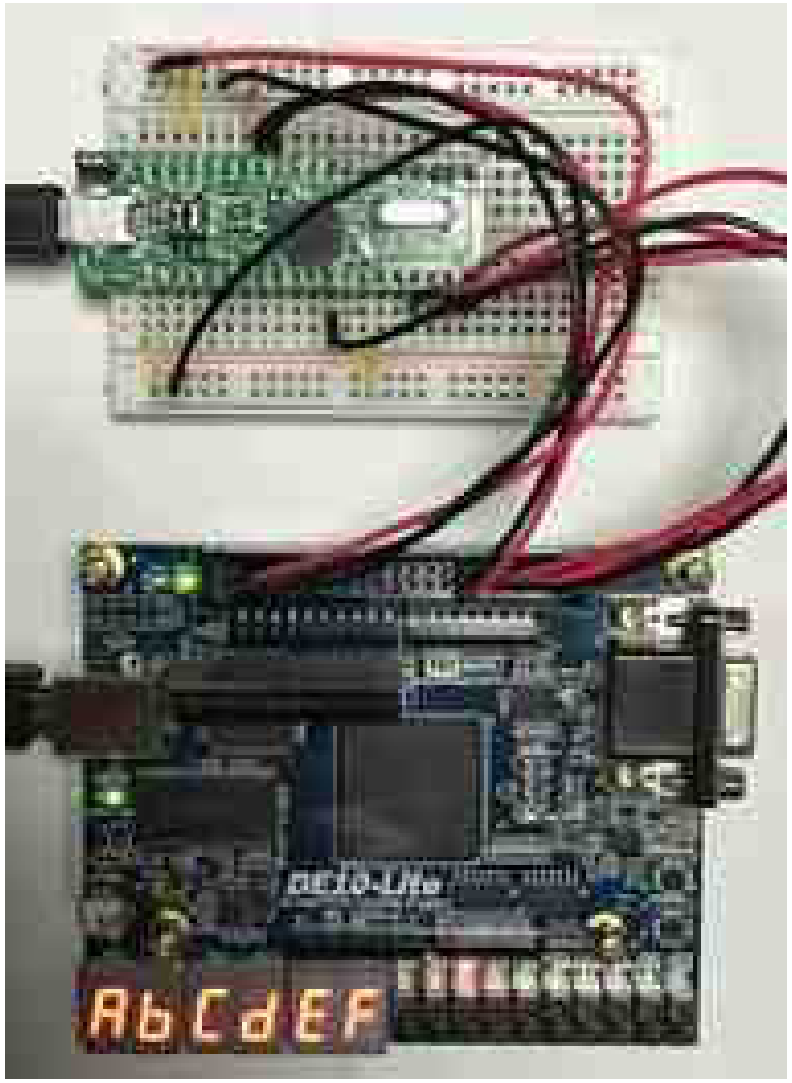
Terasic DE10-Lite Board
(Intel MAX 10 10M50DAF484C7G)

USB-JTAG

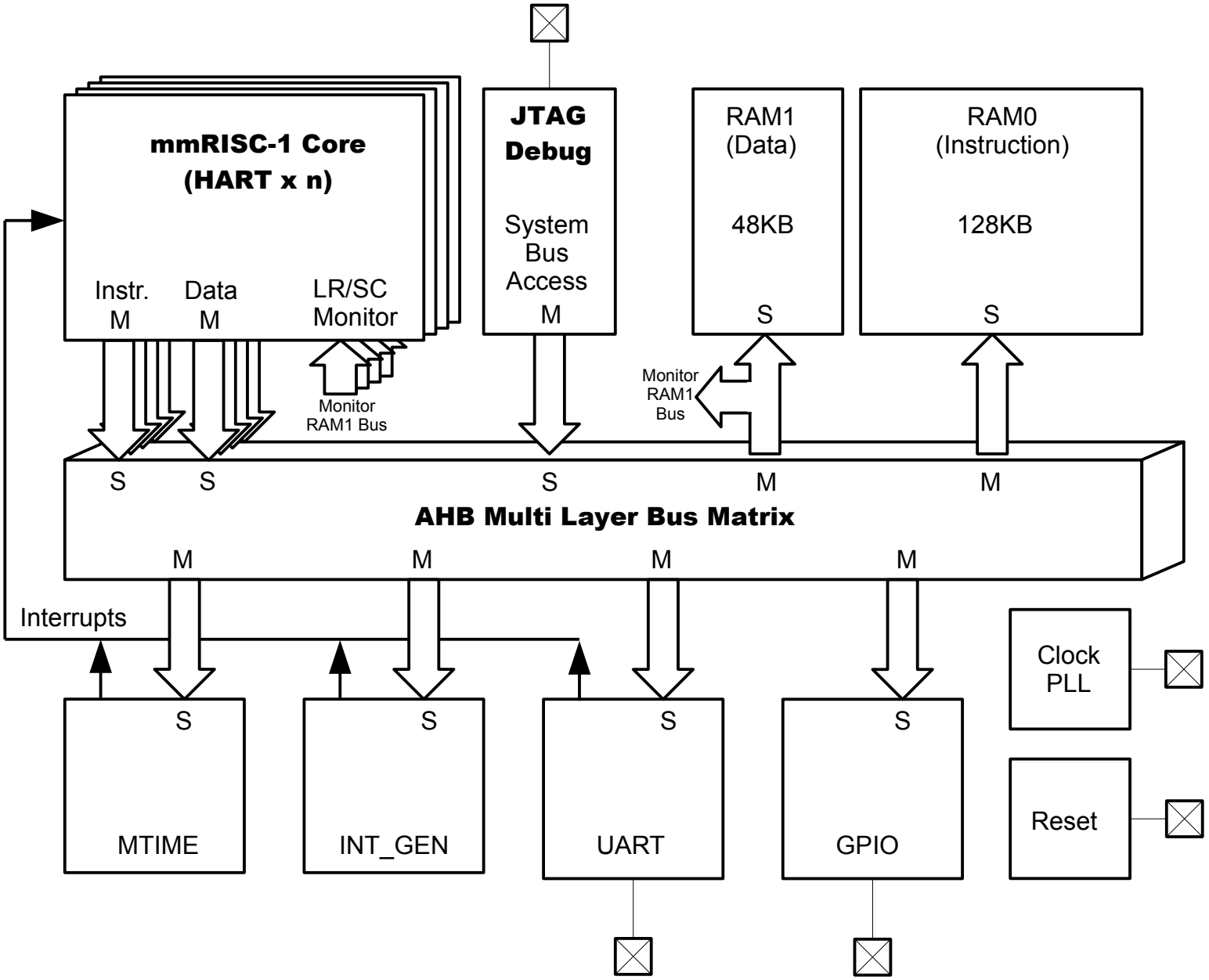
Olimex ARM-USB-OCD Compatible

< $20





Tiny SoC for FPGA System



Verification Methods

1. **RTL Verification by Vector Simulation**

2. **Random Wait Cycle in Instruction and Data Access**

3. **Bus Conflicts among Multi Harts**

4. **Verification of C Model for non-restoring Division / Remainder**
   Check many corner cases in signed/unsigned values.

5. **Verification of C Model for Floating Point using GNU Scientific Library**
   Combination of Zero / INF / QNAN / SNAN / Subnormal / Normal / Sign
   Rounding Mode : RNE / RTZ / RUP / RDN, x4 cases
   FADD/FSUB/FMULFDIV : 4 x (Sign 2 x Expo 256 x Frac 24) ^ 2 = 604M cases
   FSQRT : 4 x (Sign 2 x Expo 256 x Frac 24) = 49K cases

6. **RTL Verification using GitHub “riscv-arch-test” (ISA IMC, privileged, zifence)**

7. **RTL Verification using GitHub “riscv-tests” (ISA IMACF, privileged)**

8. **Several C applications and benchmarks on FPGA system.**

FPGA Implementation Result

RISC-V Spec	RV32IMAC	RV32IMAFC
Peripherals	Instruction RAM 128KB Data RAM 48KB Multilayer BUS UART / Timer / GPIO	
Device	MAX 10 10M50DAF484C7G	
Quartus Prime	Lite Edition Ver.20.1.1	
Optimization	Balanced (Normal Flow)	
Logic Elements	23062/49760 = 46%	41589/49760 = 84%
Interconnect Usage	30%	38%
Frequency	20MHz	16.67MHz
Worst Setup Slack	3.999ns (met)	0.215ns (met)

Basic Samples for CPU and FPU

Startup Routine, Interrupt, UART, printf(), LED, Switch, ...

FreeRTOS

1. Ported FreeRTOS 202104.00 on mmRISC-1 FPGA System.
2. Operating Demo Program “Blinky”.

   Task / Queue / Mutex / Semaphore /User Interrupt

```
Hello FreeRTOS!
0: Tx: Transfer1
0: Rx: Blink1
0: Tx: Transfer2
0: Rx: Blink2
0: Tx: Transfer1
0: Rx: Blink1
…
```

Dhrystone

	riscv64-unknown-elf-gcc 10.2.0	DMIPS/MHz	Memory
Dhrystone 2.1	-O3	1.55	1cyc RAM
Dhrystone 2.1	-O3 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las -flto	3.55	1cyc RAM

Coremark

	riscv64-unknown-elf-gcc 10.2.0	Coremark/MHz	Memory
Coremark 1.0	-O3 -funroll-loops -fpeel-loops -fgcse-sm -fgcse-las	2.76	1cyc RAM

Summary

What I have done :

To Define specification of RV32IMAFC **mmRISC-1**

To Design mmRISC-1

To Verify mmRISC-1

To Implement mmRISC-1 in FPGA

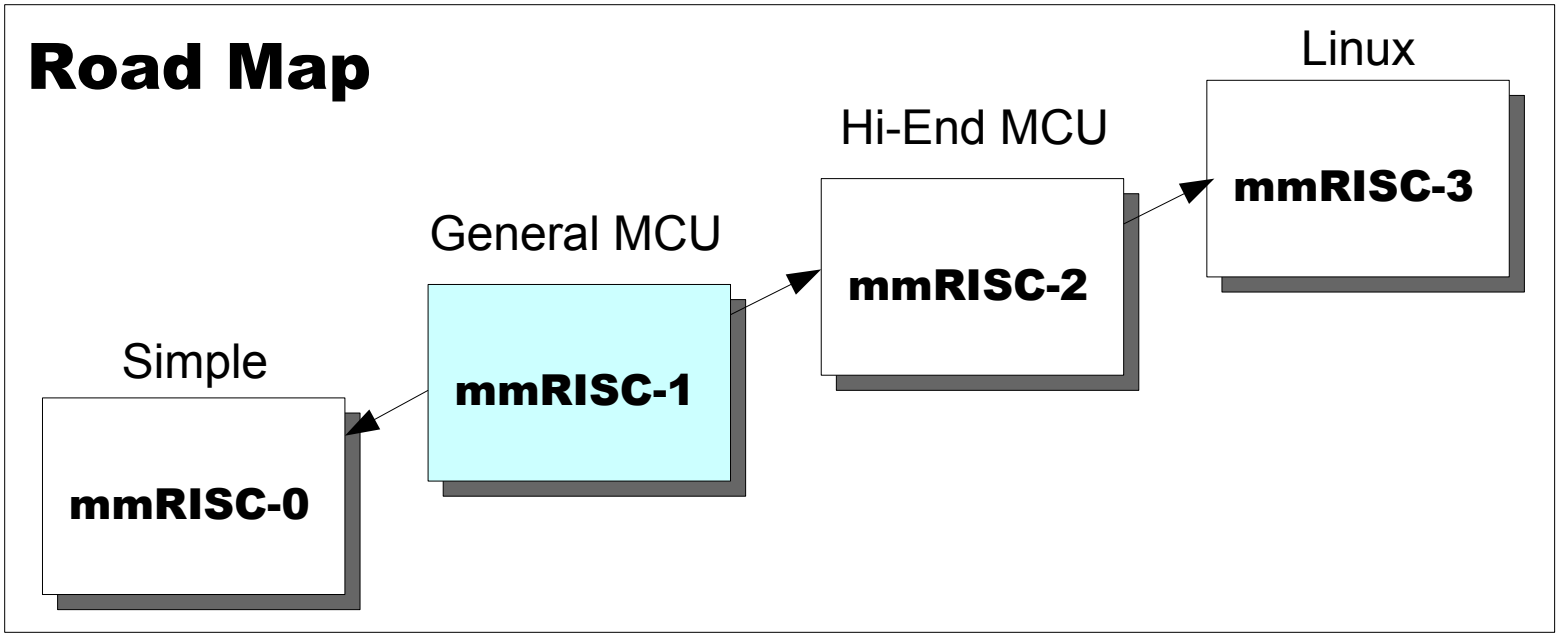
To Program some Application Softwares

To push mmRISC-1 on **Github**

`https://github.com/munetomo-maruyama/mmRISC-1`

Provided under simple 2-clause BSD License.

Details are described in
Technical Reference Manual in “doc” directory.



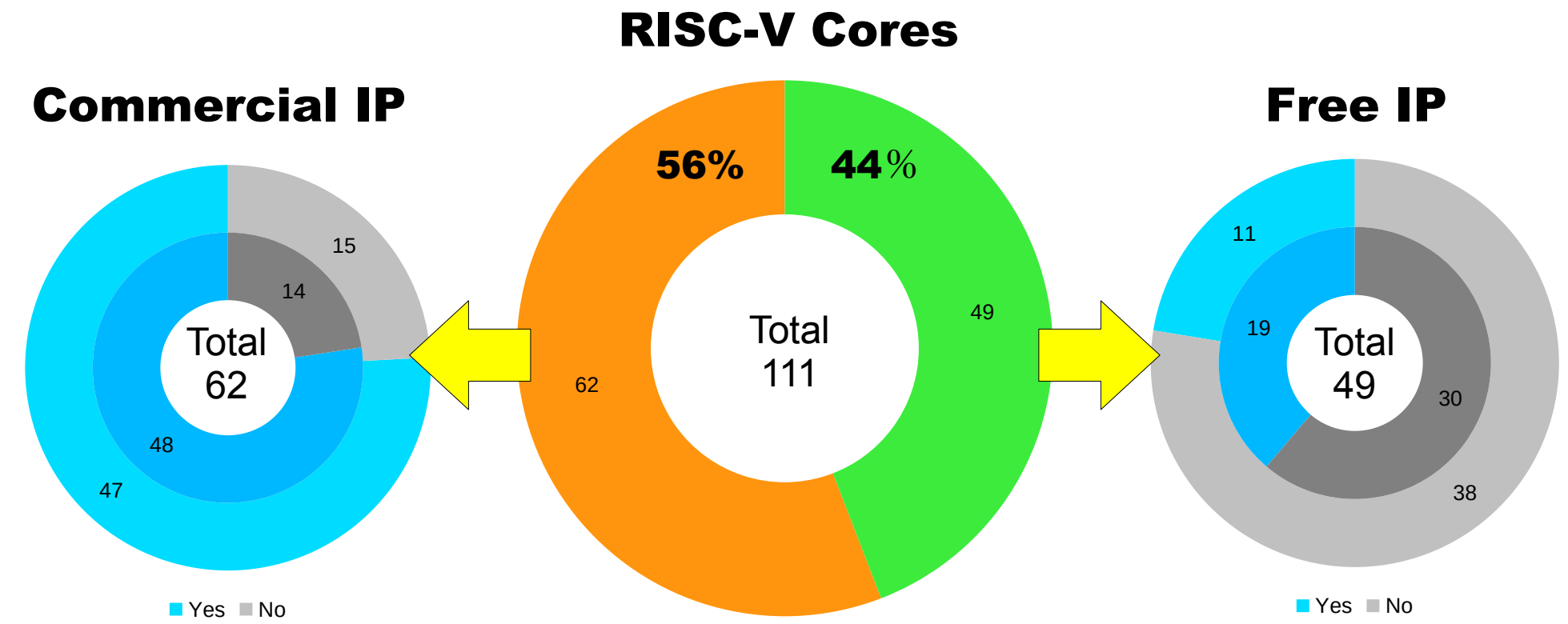
To Do

Quality Improvement and Bug Fixes
Performance Improvement
- Improve Coremark
- D Extension
- Bit Manipulation
- Custom Instruction
- Event Handling
- Low Latency Interrupt
- Branch Target Cache
- FLASH Memory Cache
- ...

Logic Size Reduction
- E Extension
- Reduce Debug Spec
- ...

Expanding Specifications
- cJTAG Debugger
- More Privileged Modes
- AXI Bus Bridge
- Functional Safety
- ...

More RISC-V Cores



Outside : Floating Supported
Inside : Atomic Supported

Outside : Floating Supported
Inside : Atomic Supported

- Commercial IP Growing

- Come on! More Free IP!

Source : RISC-V Exchange: Cores & SoCs (https://riscv.org/exchange/cores-socs/), Snap Shot on Oct.15, 2021

Thank you.