RISC-V Week Paris



Open Source Processor IP for High Volume SoCs

Rick O'Connor rickoco@openhwgroup.org

@rickoco @openhwgroup

www.openhwgroup.org



Outline



- RISC-V Introduction
 - Free & Open Instruction Set Architecture
- Challenges with SoC design and Open Source IP

- OpenHW Group & CORE-V Family
- OpenHW Group Status
 - Working Groups & Task Groups
- Summary

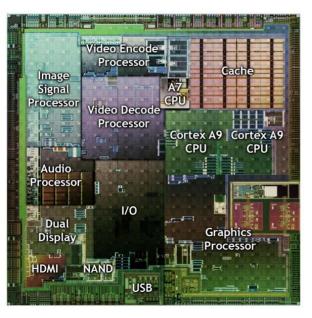




Most chips are SoCs with many ISAs



- Applications processor (usually ARM)
- Graphics processors
- Image processors
- Radio DSPs
- Audio DSPs
- Security processors
- Power-management processor
- •



NVIDIA Tegra SoC

- Apps processor ISA too large for base accelerator ISA
- IP bought from different places, each proprietary ISA
- Home-grown ISA cores
- Over a dozen ISAs on some SoCs each with unique software stack





Why so Many ISAs?



Must they be proprietary?

What if there was one free and open ISA everyone could use across all computing devices?





What's Different about RISC-V?



Simple

• Far smaller than other commercial ISAs

Clean-slate design

- Clear separation between user and privileged ISA
- Avoids µarchitecture or technology-dependent features

A modular ISA

- Small standard base ISA
- Multiple standard extensions

Designed for extensibility/specialization

- Variable-length instruction encoding
- Vast opcode space available for instruction-set extensions

RISC-V Week Paris © OpenHW Group

Stable

- Base and standard extensions are frozen
- Additions via optional extensions, not new versions





RISC-V Standard Extensions



- Four base integer ISAs
 - RV32E, RV32I, RV64I, RV128I
 - Only <50 hardware instructions needed for base
- Standard extensions
 - M: Integer multiply/divide
 - A: Atomic memory operations (AMOs + LR/SC)
 - F: Single-precision floating-point
 - D: Double-precision floating-point
 - G = IMAFD, "General-purpose" ISA
 - Q: Quad-precision floating-point
 - C: compressed 16b encodings for 32b instructions
- All the above are a fairly standard RISC encoding in a fixed 32-bit instruction format





RISC-V Reference Card

RISC-V

	TI :		<u></u>
Base Integer I	nstr	uctions (32	2 64 128)
Category Name	Fmt	RV{ 32 6	4 128)1 Base
Loads Load Byte	I	LB	rd,rs1,imm
Load Halfword	I	LH	rd,rs1,imm
Load Word	I	L{W D Q}	rd,rs1,imm
Load Byte Unsigned	I	LBU	rd,rs1,imm
Load Half Unsigned	I		rd,rs1,imm
Stores Store Byte	S	SB	rs1,rs2,imm
Store Halfword	S	SH	rs1,rs2,imm
Store Word	S	S{W D Q}	rs1,rs2,imm
Shifts Shift Left	R	SLL{ W D}	rd, rs1, rs2
Shift Left Immediate	I	SLLI{ W D}	rd, rs1, shamt
Shift Right	R	SRL{ W D}	rd, rs1, rs2
Shift Right Immediate	I	SRLI{ W D}	rd, rs1, shamt
Shift Right Arithmetic	R	SRA{ W D}	rd,rs1,rs2
Shift Right Arith Imm	I	SRAI{ W D}	rd, rs1, shamt
Arithmetic ADD	R	ADD{ W D}	rd, rs1, rs2
ADD Immediate	I	ADDI{ W D}	rd,rs1,imm
SUBtract	R	SUB{ W D}	rd, rs1, rs2
Load Upper Imm	U	LUI	rd,imm
Add Upper Imm to PC	U	AUIPC	rd,imm
Logical XOR	R	XOR	rd, rs1, rs2
XOR Immediate	I	XORI	rd,rs1,imm
OR	R	OR	rd, rs1, rs2
OR Immediate	I	ORI	rd,rs1,imm
AND	R	AND	rd, rs1, rs2
AND Immediate	I	ANDI	rd,rs1,imm
Compare Set <	R	SLT	rd, rs1, rs2
Set < Immediate	I	SLTI	rd,rs1,imm
Set < Unsigned	R	SLTU	rd, rs1, rs2
Set < Imm Unsigned	I	SLTIU	rd,rs1,imm
Branches Branch =	SB	BEQ	rs1,rs2,imm
Branch ≠	SB	BNE	rs1,rs2,imm
Branch <	SB	BLT	rs1,rs2,imm
Branch ≥	SB	BGE	rs1,rs2,imm
Branch < Unsigned	SB	BLTU	rs1,rs2,imm
Branch ≥ Unsigned	SB	BGEU	rs1,rs2,imm
Jump & Link J&L	UJ	JAL	rd,imm
Jump & Link Register	I	JALR	rd,rs1,imm
Synch Synch thread	I	FENCE	
Synch Instr & Data	I	FENCE.I	
System System CALL	I	SCALL	
System BREAK	I	SBREAK	
Counters ReaD CYCLE	I	RDCYCLE	rd
ReaD CYCLE upper Half		RDCYCLEH	rd
ReaD TIME	I	RDTIME	rd
ReaD TIME upper Half	I	RDTIMEH	rd
ReaD INSTR RETired	I	RDINSTRET	rd
ReaD INSTR upper Half	I	RDINSTRETH	rd

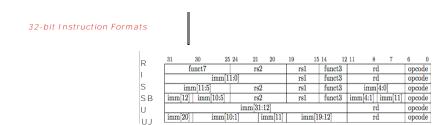
+14 Privileged

+ 8 for M

+ 34 for F, D, Q

+ 46 for C

+ 11 for A



RISC-V®

RV32I / RV64I / RV128I + M, A, F, D, Q, C



RISC-V Reference Card

Base Integer I	nstr	uctions (3.	2 64 128)	RV F	Privileged Inst	ructi	ons (32)	(64/128)	3 Opti	onal FP E	Exte	ensions: RV32{	(FIDIQ)	QO	tional Compre	ssed	Instru	ctions: RVC
Category Name	Fmt	RV(3216	54/128)/ Base	Catego	rv Name	Fmt	RV mnemo	onic	Category	Name	Fm	RV{F D Q} (I	HP/SP, DP, QP)	Catego	orv Name	Fmt		RVC
Loads Load Byte	I		rd,rs1,imm	CSR Acce		R	CSRRW	rd,csr,rs1	Load	Load	- T	FL{W,D,O}	rd,rs1,imm	Loads	Load Word	CL	C.LW	rd',rs1',imm
				II .										Luaus				
Load Halfword	I	LH	rd,rs1,imm	II .	mic Read & Set Bit		CSRRS	rd,csr,rs1	Store	Store			rs1,rs2,imm		Load Word SP		C.LWSP	rd,imm
Load Word	I	L{W D Q}	rd, rs1, imm	Atom	ic Read & Clear Bit	R	CSRRC	rd,csr,rs1	Arithmetic	ADD	R	FADD. {S D Q}	rd,rs1,rs2		Load Double	CL	C.LD	rd',rs1',imm
Load Byte Unsigned	I	LBU	rd, rs1, imm		Atomic R/W Imm	R	CSRRWI	rd,csr,imm		SUBtract	R	FSUB. {S D Q}	rd,rs1,rs2		Load Double SP	CI	C.LWSP	rd,imm
Load Half Unsigned	I	L{H W D}U	rd, rsl, imm	Atomic R	ead & Set Bit Imm	R	CSRRSI	rd,csr,imm		MULtiply	R	FMUL. {S D Q}	rd, rs1, rs2		Load Quad	CL	C.LQ	rd',rs1',imm
Stores Store Byte	S	SB	rs1,rs2,imm	Atomic Rea	d & Clear Bit Imm	R	CSRRCI	rd,csr,imm		DIVide	R	FDIV. {S D Q}	rd, rs1, rs2		Load Quad SP	CI	C.LQSP	rd,imm
Store Halfword	S	SH	rs1,rs2,imm	Change L		R	ECALL	.,,	S	Quare RooT			rd,rs1	1	oad Byte Unsigned		C.LBU	rd',rs1',imm
Store Word	S	S{W D Q}	rs1,rs2,imm	II ~	onment Breakpoint		EBREAK			Multiply-AD[_			_	Float Load Word		C.FLW	rd',rs1',imm
Shifts Shift Left	R			11		R						(- 1 - 2)						
		SLL{ W D}	rd,rs1,rs2		nvironment Return		ERET			ly-SUBtract			rd,rs1,rs2,rs3	_	Float Load Double		C.FLD	rd',rs1',imm
Shift Left Immediate	I		rd,rs1,shamt	11 '	irect to Superviso		MRTS		Negative Multip						Float Load Word SP		C.FLWSP	
Shift Right	R	SRL{ W D}	rd,rs1,rs2	II .	Trap to Hypervisor		MRTH		Negative M	ultiply-ADD	R	FMNADD. {S D Q}	rd,rs1,rs2,rs3	Flo	oat Load Double SP	CI		rd,imm
Shift Right Immediate	I	SRLI { W D}	rd, rs1, shamt	Hypervisor	Trap to Supervisor	R	HRTS		Sign Inject	SiGN source	e R	FSGNJ.{S D Q}	rd,rs1,rs2	Stores	Store Word	CS	C.SW	rs1',rs2',imm
Shift Right Arithmetic	R	SRA{ W D}	rd, rs1, rs2	Interrupt	Wait for Interrup	R	WFI		Negative S	SiGN source	R	FSGNJN. {S D Q}	rd, rs1, rs2		Store Word SP	CSS	C.SWSP	rs2,imm
Shift Right Arith Imm	I	SRAI { W D}	rd, rs1, shamt	MMU	Supervisor FENCE	R	SFENCE.VM	4 rs1	Xor 9	SiGN source	R	FSGNJX. {S D Q}	rd, rs1, rs2		Store Double	CS	C.SD	rs1',rs2',imm
Arithmetic ADD	R	ADD{ W D}	rd,rs1,rs2	Optio	nal Multiply-D	ivide	Extension	on: RV32M	Min/ Max	MINimum			rd,rs1,rs2		Store Double SP	CSS	C.SDSP	rs2,imm
ADD Immediate	I		rd,rs1,imm	Category	Name Fmt		RV32M (MAXimum			rd, rs1, rs2		Store Quad		C.SO	rs1',rs2',imm
SUBtract	R	76 78		Multiply		MIII /		d,rs1,rs2	Compare Con			(rd, rs1, rs2		Store Quad SP		_	rs2,imm
										•					•		_	*
Load Upper Imm	U	LUI	rd,imm		,	MULH		d,rs1,rs2		are Float <		FLT. {S D Q}	rd, rs1, rs2		Float Store Word		C.FSW	rd',rs1',imm
Add Upper Imm to PC	U	AUIPC	rd,imm	MULtiply H	alf Sign/Uns R	MULHS	su re	d,rs1,rs2	Comp	are Float ≤	R	FLE. {S D Q}	rd,rs1,rs2		Float Store Double		C.FSD	rd',rs1',imm
Logical XOR	R	XOR	rd,rs1,rs2	MULtiply up	per Half Uns R	MULHU	J r	d,rs1,rs2	Categorize	Classify Typ	R	FCLASS. {S D Q}	rd, rs1	F	loat Store Word SP	CSS	C.FSWSP	rd,imm
XOR Immediate	I	XORI	rd, rs1, imm	Divide	DIVide R	DIV{	W D} re	d,rs1,rs2	Move Move	from Integer	R	FMV.S.X	rd,rs1	Flo	at Store Double SP	CSS	C.FSDSP	rd,imm
OR	R	OR	rd, rs1, rs2	DIVi	de Unsigned R	DIVU	r	d,rs1,rs2	Mov	e to Integer	R	FMV.X.S	rd, rs1	Arithm	etic ADD	CR	C.ADD	rd,rs1
OR Immediate	I	ORI	rd, rs1, imm	Remainde	erREMainder R	REM{	W D} ro	d,rs1,rs2	Convert Conv	ert from In	1 R	FCVT. {S D Q}.W	rd,rs1		ADD Word	CR	C.ADDW	rd',rs2'
AND	R	AND	rd, rs1, rs2					d.rs1.rs2	Convert from I				III rd.rs1		ADD Immediate		C.ADDI	rd,imm
					nal Atomic Ins												C.ADDIW	
AND Immediate	1	ANDI	rd,rs1,imm	Category		_		28}A (Atomic)		nvert to Int					ADD Word Imm			rd,imm
Compare Set <	R	SLT	rd, rs1, rs2			_		- ' - '	Convert to I				!} ra,rsi		ADD SP Imm * 16			6SP x0,imm
Set < Immediate	I	SLTI	rd,rs1,imm			-	W D Q}	rd,rs1	Configuratio	n Read Stat	R	FRCSR	rd		ADD SP Imm * 4		C.ADDI4	SPN rd',imm
Set < Unsigned	R	SLTU	rd, rs1, rs2	Store Sto	re Conditiona R	SC.{	W D Q}	rd,rs1,rs2	Read Rou	nding Mode	R	FRRM	rd		Load Immediate	CI	C.LI	rd,imm
Set < Imm Unsigned	I	SLTIU	rd, rs1, imm	Swap	SWAP R	AMOS	WAP. {W D	Q} rd,rs1,rs2		Read Flags	R	FRFLAGS	rd		Load Upper Imm	CI	C.LUI	rd,imm
Branches Branch =	SB	BEQ	rs1,rs2,imm	Add	ADD R	AMOA	DD. (W D Q	} rd,rs1,rs2	Swap	Status Reg	R	FSCSR	rd,rs1		MoVe	CR	C.MV	rd,rs1
Branch ≠	SB	_	rs1,rs2,imm	Logical	XOR R	AMOX	OR. {W D Q			nding Mode			rd, rs1		SUB		C.SUB	rd',rs2'
Branch <	SB				AND R		ND. {W D Q			Swap Flags		FSFLAG	•		SUB Word		C.SUBW	rd',rs2'
			rs1,rs2,imm		OR R		R.{W D Q}						rd,rs1					
Branch ≥	SB		rs1,rs2,imm	Min/May	MINimum R				Swap Rounding		1	FRMI U	rd,imm	Logica			C.XOR	rd',rs2'
Branch < Unsigned	SB		rs1,rs2,imm	Pilli/ Plax			IN. {W D Q		Cwar	Flane Imm	1 1	FSFT.AGST	rd.imm		OR		C.OR	rd',rs2'
Branch ≥ Unsigned	SB	BGEU	rs1,rs2,imm	4	MAXimum R			} rd,rs1,rs2					S 7 /		AND		C.AND	rd',rs2'
Jump & Link J&L	UJ	JAL	rd,imm	MINim	um Unsigned R			Q} rd,rs1,rs2		./ []]		11)	11/		AND Immediate	CB	C.ANDI	rd',rs2'
Jump & Link Register	I	JALR	rd,rs1,imm	MAXimi	um Unsigned R	AMOM	AXU. {W D	Q} rd,rs1,rs2		l (th		D C	L (Shifts	Shift Left Imm	CI	C.SLLI	rd,imm
Synch Synch thread	I	FENCE										1 1	737	Shi	ft Right Immediate	СВ	C.SRLI	rd',imm
Synch Instr & Data	т	FENCE.I									_		_		ift Right Arith Imm		C.SRAI	rd',imm
System System CALL	I	SCALL		1					1	20		F D	\cap 1	Branch			C.BEOZ	rs1',imm
	T .			16-hit / D	VC) and 32-bit	Inetr	uction For	rmate		70	1	TIリー	U) }	וטוומוט			_	•
System BREAK	I	SBREAK		TO-DIL (K	voj anu 32-bit	111511	action FUI	rmats	_		ſ,	. _	~)		Branch≠0		C.BNEZ	rs1',imm
Counters ReaD CYCLE	I	RDCYCLE	rd	15 14	13 12 11 10 9 8	7 6	5 4 3 2	1 0				_		Jump	Jump		C.J	imm
ReaD CYCLE upper Half	I	RDCYCLEH	rd		nct4 rd/rs1		rs2	on	00 00 0	04 00					Jump Register	CR	C.JR	rd,rs1
ReaD TIME	I	RDTIME	rd	CSS funct			imm	on P 01	30 25 24 funct7	21 20 rs2	19 rs		8 7 6 0 rd opcode	Jump 8	& Link J&L	CJ	C.JAL	imm
ReaD TIME upper Half	I	RDTIMEH	rd	CI W funct				op	imm[11:0]	182	rs		rd opcode	Ju	mp & Link Register	CR	C.JALR	rs1
ReaD INSTR RETired	I	RDINSTRET	rd	CL funct				op S in	ım[11:5]	rs2	rs		n[4:0] opcode	System	n Env. BREAK	CI	C.EBREA	K
ReaD INSTR upper Half	ī	RDINSTRETH		CS funct		imi		op	imm[10:5]	rs2	rs		imm[11] opcode	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,				
				_ Iunict	num LSI	1 11/11	152	op SB mm[12	[-510]		-0	111						

funct3



RV32I / RV64I / RV128I + M, A, F, D, Q, C RISC-V "Green Card"

₹ <mark>7</mark> F	₹ I:	SC-\	/											RISC	-V Ref	er	ence	Card
Base Integer	Inst	ructions (3	21641128)	RV Priv	ileaed Insi	ruct	ions (321	641128)	3 Ont	ional FP Fx	(te	nsions: RV32	{ FI DI O}	Ontio	nal Compre.	ssen	IInstruc	rtions: RVC
Category Name	Fm		64/128)/ Base	Category	Name	Fmt	RV mnemo		Category	Name F	mı		HP/SP, DP, QP)	Category	Name	Fmt	77751740	RVC
Loads Load Byte	I	LB	rd,rs1,imm	CSR Access	Atomic R/W	R	CSRRW	rd,csr,rs1	Load	Load	I	FL{W,D,O}	rd,rs1,imm	Loads	Load Word	CL	C.LW	rd',rs1',imm
Load Halfword		LH	rd,rs1,imm		Read & Set Bit		CSRRS	rd,csr,rs1			$\overline{}$	FS{W,D,Q}	rs1,rs2,imm		Load Word SP			rd,imm
Load Word		T{MIDIO}	rd, rs1, imm		ad & Clear Bit		CSRRC	rd,csr,rs1	Arithmetic		-	FADD. {S D O}	rd, rs1, rs2		Load Double	-		rd',rs1',imm
Load Byte Unsigned		LBU	rd, rs1, imm		mic R/W Imm	R	CSRRWI	rd,csr,imm			- 1	FSUB. {S D O}	rd, rs1, rs2		Load Double SP	-		rd,imm
Load Half Unsigned		L{H W D}U	rd,rs1,imm	Atomic Read 8		R	CSRRSI	rd,csr,imm			- 1	FMUL. {S D O}	rd, rs1, rs2		Load Quad			rd',rs1',imm
Stores Store Byte	_	SB	rs1,rs2,imm	Atomic Read & 0		R	CSRRCI	rd,csr,imm			- 1	FDIV. (S D Q)	rd, rs1, rs2		Load Quad SP		_	rd,imm
Store Halfword		SH	rs1,rs2,imm	Change Level		R	ECALL	20/001/21				FSQRT.{S D Q}		Load	Byte Unsigned			rd',rs1',im
Store Word		S{W D O}	rs1,rs2,imm		nt Breakpoint	R	EBREAK			-	\rightarrow	FMADD.{S D Q}		1	loat Load Word			rd',rs1',im
Shifts Shift Left	R		rd,rs1,rs2	-11	nment Return	R	ERET						rd,rs1,rs2,rs3		at Load Double			rd',rs1',im
Shift Left Immediate			rd,rs1,shamt	Trap Redirect			MRTS		Negative Multi				rd,rs1,rs2,rs3		t Load Word SP		C.FLWSP	
Shift Right		SRL{ W D}		Redirect Trap			MRTH		-			FMNADD. {S D Q}			Load Double SP		C.FLDSP	
Shift Right Immediate				Hypervisor Trap			HRTS				\rightarrow	FSGNJ.{S D Q}		Stores	Store Word			rs1',rs2',i
Shift Right Arithmetic			rd, rs1, rs2	Interrupt Wa		R	WFI		11 ~ ,			FSGNJN. (S D Q)		0.0103	Store Word SP			rs2,imm
Shift Right Arith Imm	I		rd,rs1,shamt		ervisor FENCE		+	rs1	11 -			FSGNJX. {S D Q			Store Double			rs1',rs2',i
Arithmetic ADD	R	1	rd, rs1, rs2				e Extensio		Min/ Max		\rightarrow	FMIN. {S D Q}	rd, rs1, rs2		tore Double SP			rs2,imm
ADD Immediate			rd,rs1,imm	Category	Name Fmt	TVTG	RV32M (N		min max			FMAX. {S D O}	rd, rs1, rs2		Store Quad		C.SQ	rs1',rs2',i
SUBtract			rd, rs1, rs2		4ULtiply R	MIII. 8		l,rs1,rs2	Compare Co i		\neg	FEQ. {S D Q}	rd, rs1, rs2		Store Quad SP			rs2,imm
Load Upper Imm			rd,imm	MULtiply up		MULF		l,rs1,rs2		-		FLT. {S D O}	rd, rs1, rs2	FI	oat Store Word			rd',rs1',im
Add Upper Imm to PC	: U		rd,imm	MULtiply Half S		MULH		l,rs1,rs2		pare Float ≤			rd, rs1, rs2		at Store Double			rd',rs1',im
_ogical XOR	R		rd,rs1,rs2	MULtiply upper I	3 ,	MULF		l,rs1,rs2				FCLASS. (SIDIO)			Store Word SP		C.FSWSP	
XOR Immediate		XORI	rd,rs1,imm	Divide	DIVide R			l,rs1,rs2	, ,	, , , , , ,	\rightarrow	FMV.S.X	rd,rs1	-	Store Double SP			- *
OR Illillediate	R	OR	rd, rs1, rs2	DIVide U		DIV		l,rs1,rs2		~	٠,١	FMV.X.S	rd, rs1	Arithmeti			C.ADD	rd,rs1
OR Immediate	1	ORI	rd,rs1,imm	RemainderRE	noigned it			l,rs1,rs2	11	vert from Int		FCVT. {S D Q}.V		Arminen	ADD Word	_	C.ADDW	rd',rs2'
AND		AND	rd, rs1, rs2	REMainder U				l,rs1,rs2	Convert from 1			FCVT. {S D Q}.			ADD Word		C.ADDI	rd,imm
AND Immediate		ANDI	rd,rs1,imm		- 5 1			nsion: RVA	il			FCVT.W. {S D Q}			ADD Word Imm	-	C.ADDIW	rd,imm
Compare Set <	R	1	rd, rs1, rs2	Category	Name Fmt			8} A (Atomic)	1			FCVT.WU.{S D Q			D SP Imm * 16	-		SP x0,imm
Set < Immediate		SLTI	rd,rs1,imm	Load Load Re		_	(WIDIO)	rd,rs1	Configuration		\rightarrow	FRCSR	rd		DD SP Imm * 4			
Set < Unsigned			rd, rs1, rs2	Store Store Co		-	(WIDIQ)	rd,rs1,rs2	11 ~		- 1	FRRM	rd		oad Immediate		C.LI	rd,imm
Set < Imm Unsigned	T	STITU	rd,rs1,imm	Swap	SWAP R	-)} rd,rs1,rs2	Read Rot			FRFLAGS	rd		oad Immediate oad Upper Imm		C.LUI	rd,imm
Branches Branch =	SB		rs1,rs2,imm	Add	ADD R		ADD. (WIDIO)		Cwa	-	- 1	FSCSR	rd, rs1		MoVe		C.MV	rd, rmun
Branch #		BNE	rs1,rs2,imm	Logical	XOR R		KOR.{W D Q}		11		- 1	FSRM	rd,rs1		SUB		C.SUB	rd',rs2'
Branch <	SB		rs1,rs2,imm	Logical	AND R			rd, rs1, rs2	Swap Kui			FSFLAGS	rd,rs1		SUB Word		C.SUBW	rd',rs2'
Branch ≥			rs1,rs2,imm		OR R		AND. {W D Q} OR. {W D O}	rd, rs1, rs2	Swap Roundin		- 1	FSRMI	rd, rsi rd, imm	Logical	XOR	_	C.XOR	rd',rs2'
Branch < Unsigned			rs1,rs2,imm	Min/Max MI	Nimum R		MIN. {W D Q}		11	p Flags Imm		FSFLAGSI	rd,imm	Logical	OR		C.XOR	rd',rs2'
Branch ≥ Unsigned			rs1,rs2,imm		AXimum R		41N. (W D Q) 4AX. (W D Q)				_	ons: RV{ 64 1.	·		AND		C.AND	rd',rs2'
Jump & Link J&L	_	JAL	rd,imm	MINimum U				rd, rs1, rs2	Category	Name F	$\overline{}$	RV{ F D Q} (,	AND Immediate		C.ANDI	rd',rs2'
Jump & Link Register	. 03	JALR	rd, imm rd, rs1, imm	MAXimum U	-)} rd,rs1,rs2)} rd,rs1,rs2	Move Move			FMV. {D O}.X	rd, rs1	 	Shift Left Imm	-	C.SLLI	rd, rsz.
Synch Synch thread	I	1	ru, rsi, inni	MAXIIIIIIII U	naigheu K	AMON	MAD. [WIDIQ	// ru, rsr, rsz	11	~	٠,١	FMV.X.{D Q}	rd,rs1		ight Immediate		C.SELI	rd, imm
Synch Instr & Data	T	FENCE.I							Convert Cor			FCVT. {S D Q}.			Right Arith Imm		C.SRAI	rd',imm
System System CALL	I	SCALL							Convert from 1			FCVT. {S D Q}.		Branches		-	C.BEOZ	ra',ımm rs1',imm
System CALL System BREAK	_	SBREAK		16-bit (RVC)	and 32-hit	Insti	ruction For	mats				FCVT. {L T}. {S		branciles	Branch=0 Branch≠0	-	C.BEQZ C.BNEZ	rs1',imm
System BREAK Counters ReaD CYCLE		RDCYCLE	rd	- Dir (1.00)	3 G 52 DII	. 11511	. 3311011 1 01		Convert to 1			FCVT. {L T}U. {S		Jump	<u>Brancn≠u</u> Jump	-	C.BNEZ	imm
ReaD CYCLE upper Hal		RDCYCLEH	ra rd	CI 15 14 13 1		7 6	5 4 3 2 1		Convert to I	inc onsigned [K	10.11.[1]1]0.[1	7,2,2, 10,131	Julip	Jump Register		C.JR	
ReaD CYCLE upper Hai		RDCYCLEH	rd rd	Tuncta	rd/rs1			op 31	30 25 24	21 20 19		15 14 12 11	8 7 6 0	Jump & Li		_	C.JAL	rd,rs1 imm
ReaD TIME upper Hal		RDTIME	ra rd	CSS funct3 im	m rd/rs1			op R	funct7		rs1		rd opcode		& Link Register		C.JALR	rs1
ReaD TIME upper Hai		RDINSTRET		funct3	imm		rd'	op -	imm[11:0]		rs1		rd opcode	System	Env. BREAK		C.EBREAR	
ReaD INSTR RETIFECT		RDINSTRET		CL funct3	imm rs1		nm rd'	op S 11	nm[11:5]		rs1		m[4:0] opcode imm[11] opcode	System	LIIV. BKEAK	CI	C.EBKEAF	
vean TNOTK upper Hal	1 1	RDINSTRETH	1 LQ	CS funct3	imm rs1 offset rs1			op .		nm[31:12]	151		rd opcode					
				CB funct3	OHSEL ISI			op U imm 20		imm[11]	im		rd opcode					

Outline



- RISC-V Introduction
 - Free & Open Instruction Set Architecture
- Challenges with SoC design and Open Source IP
- OpenHW Group & CORE-V Family
- OpenHW Group Status
 - Working Groups & Task Groups
- Summary



SoC Development Cost Drivers



- Software, RTL design, Verification and Physical design account for ~90% of overall SoC development costs
- For highly differentiated IP blocks and functions, this investment is warranted
- For general purpose CPU cores an effective open-source model can drive down these development costs and increase re-use across the industry

RISC-V Week Paris © OpenHW Group

SoC development cycle cost (\$%)

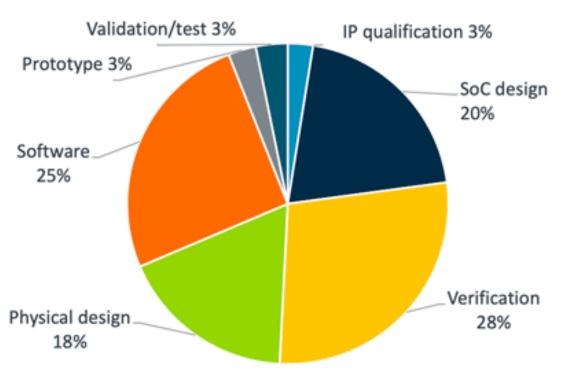


Image Source: Arm Ecosystem Blog



What problem are we trying to solve?



Barriers to adoption of open-source cores

- IP quality
 - harness community best-in-class design and verification methods and contributions
- Ecosystem
 - ensure availability of IDE, RTOS / OS ports, physical design etc. and create a roadmap of cores covering a range of PPA metrics
- Permissive use
 - permissive open-source licensing and processes to minimize business and legal risks



RISC-V ISA Brings Open Source Paradigm to CPU Design



- The free and open RISC-V ISA unleashes a new frontier of processor design and innovation
- How many open source processor implementations do we need as an industry?
 - Open cores are great from a pedagogical teaching perspective, but how many is too many for widespread industry adoption?
- How does the industry, ecosystem, community organize to ensure open core success?

RISC-V Week Paris © OpenHW Group

• How do we establish critical mass around a handful of open cores?



Many RISC-V Open Source Cores... ...and

RISC-V Week Paris © OpenHW Group

counting



COL	inting.	• • •			(source: riscv	v.org)
Name	Maintainer	Links	User spec	License	Name	Main
rocket	SiFive, UCB Bar	<u>GitHub</u>	2.3-draft	BSD	Minerva	Lambda
freedom	SiFive	<u>GitHub</u>	2.3-draft	BSD	OPenV/mriscv	OnChip
Berkeley Out-of-		<u>GitHub</u>	2.3-draft	BSD	VexRiscv	SpinalH
Order Machine (BOOM)	Bar				Roa Logic RV12	Roa Log
ORCA	VectorBlox	<u>GitHub</u>	RV32IM	BSD	SCR1	Cuntaco
RI5CY	ETH Zurich, Università di Bologna	<u>GitHub</u>	RV32IMC	Solderpad Hardware License v. 0.51		Syntaco
Zero-riscy	ETH Zurich, Università di Bologna	<u>GitHub</u>	RV32IMC	Solderpad Hardware License v. 0.51	E200 Shakti	Bob Hu IIT Mad
Ariane	ETH Zurich, Università di	Website, GitHub	RV64GC	Solderpad Hardware	ReonV	Lucas Ca
	Bologna			License v. 0.51		Clifford
Riscy Processors	MIT CSAIL CSG	Website, GitHub		MIT	MR1	Tom Ve
RiscyOO	MIT CSAIL CSG	GitHub	RV64IMAFD	MIT	SERV	
,					SweRV EH1	Western
Lizard	Cornell CSL BRG	<u>GitHub</u>	RV64IM	BSD	D D	Corpora

Name	Maintainer	Links	User spec	License
Minerva	LambdaConcept	<u>GitHub</u>	RV32I	BSD
OPenV/mriscv	OnChipUIS	<u>GitHub</u>	RV32I(?)	MIT
VexRiscv	SpinalHDL	<u>GitHub</u>	RV32I[M][C]	MIT
Roa Logic RV12	Roa Logic	<u>GitHub</u>	2.1	Non-Commercial License
SCR1	Syntacore	<u>GitHub</u>	2.2, RV32I/E[MC]	Solderpad Hardware License v. 0.51
Hummingbird E200	Bob Hu	<u>GitHub</u>	2.2, RV32IMAC	Apache 2.0
Shakti	IIT Madras	Website, GitLab	2.2, RV64IMAFDC	BSD
ReonV	Lucas Castro	<u>GitHub</u>		GPL v3
PicoRV32	Clifford Wolf	<u>GitHub</u>	RV32I/E[MC]	ISC
MR1	Tom Verbeure	<u>GitHub</u>	RV32I	Unlicense
SERV	Olof Kindgren	<u>GitHub</u>	RV32I	ISC
SweRV EH1	Western Digital Corporation	<u>GitHub</u>	RV32IMC	Apache 2.0
Reve-R	Gavin Stark	<u>GitHub</u>	RV32IMAC	Apache 2.0



Outline



- RISC-V Introduction
 - Free & Open Instruction Set Architecture
- Challenges with SoC design and Open Source IP

- OpenHW Group & CORE-V Family
- OpenHW Group Status
 - Working Groups & Task Groups
- Summary









 OpenHW Group is a not-for-profit, global organization driven by its members and individual contributors where HW and SW designers collaborate in the development of open-source cores, related IP, tools and SW such as the CORE-V Family of cores. OpenHW provides an infrastructure for hosting high quality open-source HW developments in line with industry best practices.





































CORE-V[™] Family of RISC-V Cores



- Initial contribution of open source RISC-V cores from <u>ETH Zurich PULP</u> Platform
- ETH zürich
 Integrated Systems Laboratory



- Very popular, industry adopted cores
- OpenHW Group becomes the official committer for these repositories



Core	Bits/Stages	Description
RI5CY	32bit / 4-stage	A 4-stage core that implements, the RV32IMFCXpulp, has an optional 32-bit FPU supporting the F extension and instruction set extensions for DSP operations, including hardware loops, SIMD extensions, bit manipulation and post-increment instructions.
<u>Ariane</u>	64bit / 6-stage	A 6-stage, single issue, in-order CPU implementing RV64GC extensions with three privilege levels M, S, U to fully support a Unix-like (Linux, BSD, etc.) operating system. It has configurable size, separate TLBs, a hardware PTW and branch-prediction (branch target buffer, branch history table and a return address stack).

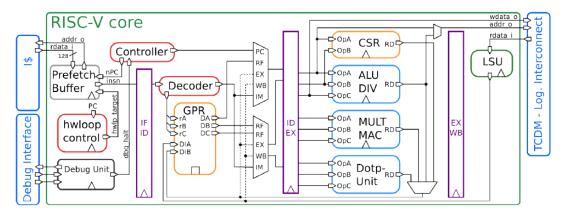




CORE-V[™] CV32 Processor IP



- 4-stage pipeline
 - RV32IMFCXpulp
 - 70K GF22 nand2 equivalent gate (GE) + 30KGE for FPU
 - Coremark/MHz 3.19
 - Includes various extensions
 - pSIMD
 - Fixed point
 - Bit manipulations
 - HW loops
- Silicon Proven
 - SMIC130, UMC65, TSMC55LP, TSMC40LP, GF22FDX



- Floating Point Unit
 - Iterative DIV/SQRT (9 cycles)
 - Parametrizable latency for MUL, ADD, SUB, Cast
 - Single cycle load, store

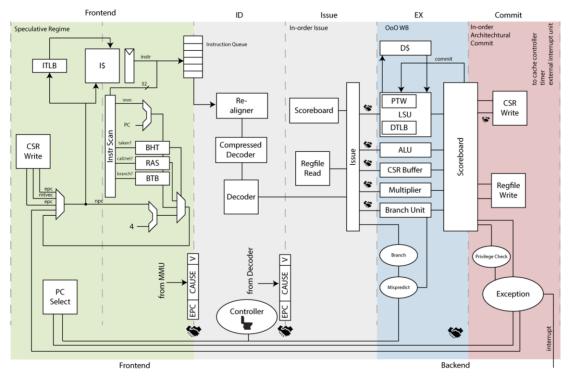




CORE-V[™] CV64 Processor IP



- Application class 64bit RV64GC processor core IP
- Linux Capable
 - Tightly integrated D\$ & I\$
 - M, S & U privilege modes
 - TLB, SV39
 - Hardware PTW
- Optimized for performance
 - Frequency: 1.5GHz (22FDX)
 - Area:~175kGE
 - Critical path: ~25logiclevels



6-stage pipeline

- In-order issue
- Out-of-order write-back
- In-order commit





Proven Processor IP



- Honey Bunny 2015 GF 28nm
- Imperio 2015 UMC 65nm
- Patronus 2016 UMC 65nm
- Mr. Wolf 2017 TSMC 40nm
- **Arnold** 2018 GF 22FDX
- Poseidon 2018 GF 22FDX
- <u>Drift</u> 2018 UMC 65nm
- Atomario 2018 UMC 65nm
- Kosmodrom 2018 GF 22FDX

- Scarabaeus 2018 UMC 65nm
- <u>VivoSoc3</u> 2018 SMIC 110nm
- Baikonur 2019 GF 22FDX
- PLINK 2019 UMC 65nm
- **Urania** 2019 UMC 65nm
- Xavier 2019 UMC 65nm
- GreenWaves GAP8
- NXP Vega Board



GREENWAVES

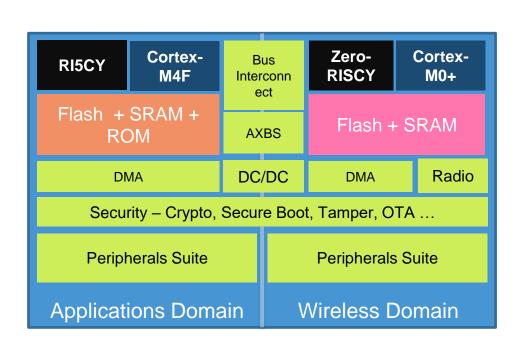




CORE-V[™]CR32 NXP VEGA Board



- Main applications running on CORE-V (RI5CY) core
- Zephyr, Micropython + drivers, all upstreamed and available on Github







Outline



- RISC-V Introduction
 - Free & Open Instruction Set Architecture
- Challenges with SoC design and Open Source IP
- OpenHW Group & CORE-V Family
- OpenHW Group Status
 - Working Groups & Task Groups
- Summary



OpenHW Group Board of Directors



Five member board of directors

Initial BoD appointed with staggered term

RISC-V Week Paris © OpenHW Group

Replacements elected by membership



- Charlie Hauck, Bluespec (Treasurer)
- Alessandro Piovaccari, Silicon Labs
- Xiaoning Qi, Alibaba Group
- Rick O'Connor, OpenHW Group











OpenHW Group structure



- On behalf of the membership, Board of Directors responsible for fulfilling the organization's purpose
- Board appoints Chairs of working groups and has final approval of working group recommendations
 - Technical Working Group and Marketing Working Group will be standing working groups
- All working group participants must be organization members
- WG Chairs report to the Board
- WGs are subject to termination if not making satisfactory progress



Working Groups & Task Groups



 Board appoints Chairs of ad-hoc working groups and has final approval of working group recommendations

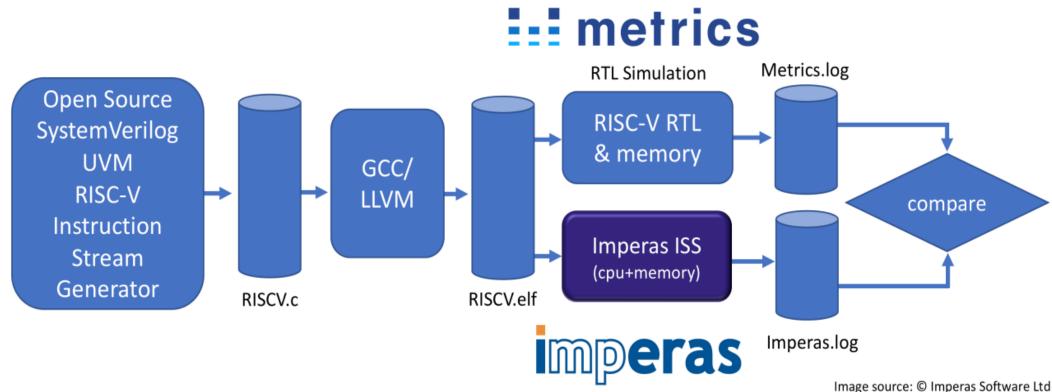
- Technical Working Group and Marketing Working Group will be standing working groups
- Technical Working Group
 - Cores Task Group
 - Verification Task Group
 - Platform Task Group
- Marketing Working Group
 - Content Task Group
 - Events Task Group



Verification Task Group



- Cloud Based Open Source Verification Test Bench
- All test stimulus (directed, random, compliance) is open-source and optimized to run on commercial System Verilog simulators (subsets can also run on Verilator)





OpenHW Group Status



- Legally registered open source, not-for-profit corporation
 - International footprint with developers in North America, Europe and Asia
- OpenHW Group & CORE-V Family launched June 6th, 2019
 - Visit www.openhwgroup.org for further details
- Follow us on Social media
 - Twitter <u>@openhwgroup</u>
 - LinkedIn OpenHW Group
- Strong <u>supporting testimonials</u> from 20+ sponsors & partners

RISC-V Week Paris © OpenHW Group

• Join our mailing list to learn how you can influence the shape of to this important open source hardware initiative.









- Strong support from industry, academia and individual contributors
- Proven System Verilog core designs, processor sub-system IP blocks, verification test bench, and reference designs
- Industry proven IDEs, a wide range of RTOS/OS ports and extensive libraries to build necessary software stacks
- Validated EDA tool flows and proven PPA characteristics
- International footprint with developers in North America, Europe and Asia



































