TWG Opcode Issues

ISA Extensions to be Supported by the Tool Chain
Instruction Encoding for CORE-V Instructions

Jeremy Bennett jeremy.bennett@embecosm.com

T @jeremypbennett T @openhwgroup

www.openhwgroup.org

Issues to Address

- Which CORE-V ISA extensions to support in the tool chain
- Encoding of the supported ISA extensions

PULP ISA Extensions

- Post-incrementing load/store (25)
  - includes register indexing
- Hardware loops (6)
- General ALU operations (31)
  - assorted specialist arithmetic operations
- Immediate branching operations (2)
  - comparison between register and 5-bit immediate
- Multiply-accumulate (22)
- Bit manipulation (16)
- SIMD (220)

$\}$ Being standardized by RISC-V International B and P task groups

Proposal for CORE-V SW Tools

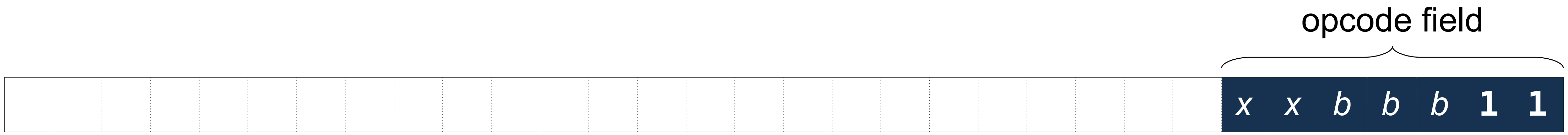
- CORE-V tools to support the following ISA extensions
  - post-increment load/store
  - hardware loops
  - general ALU operations
  - immediate branching
  - multiply accumulate
- CORE-V tools to *not* support the following ISA extensions
  - bit manipulation
  - SIMD
- This results in supporting 86 instructions rather than 322
  - the omitted functionality will be available as standard in due course

CORE-V Instruction Encoding

- Current PULP ISA extensions collide with RISC-V encodings
  - an absolute blocker to upstreaming the tool chains

- Examples
  - bit manipulation uses **OP** space
  - SIMD uses *reserved* opcode space
  - some general ALU instructions use **OP** space

RISC-V 32-bit Opcode Groups



bbb	000	001	010	011	100	101	110	111
xx								(>32*b*)
00	LOAD	LOAD-FP	*custom-0*	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48*b*
01	STORE	STORE-FP	*custom-1*	AMO	OP	LUI	OP-32	64*b*
10	MADD	MSUB	NMSUB	NMADD	OP-FP	*reserved*	*custom-2*	48*b*
11	BRANCH	JALR	*reserved*	JAL	SYSTEM	*reserved*	*custom-3*	≥ 80*b*

custom-2 and *custom-3* are also reserved for **RV128**

RISC-V 32-bit Instruction Formats

Formatf	Bit																															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register/register	funct7							rs2					rs1					funct3			rd					opcode						
Immediate	imm[11:0]												rs1					funct3			rd					opcode						
Upper immediate	imm[31:12]																				rd					opcode						
Store	imm[11:5]							rs2					rs1					funct3			imm[4:0]					opcode						
Branch	imm[12,10:5]							rs2					rs1					funct3			imm[4:1,11]					opcode						
Jump	imm[20,10:1,11,19:12]																				rd					opcode						
Branch imm. (CV)	imma[12,10:5]							immb[4:0]					rs1					funct3			imma[4:1,11]					opcode						

Proposal for Instruction Encoding

- Encode CORE-V supported extensions in *custom* space
  - try to avoid *custom-2* and *custom-3* if possible

- Leave SIMD and bit manipulation unchanged
  - these will be replaced by standard SIMD and bitmanip in future

- Current proposal in **`cv32e40p`** repository issue 452
  - prepared by John Martin
  - MAC and general ALU operations need a lot of encoding space
    - probably impossible to avoid *custom-2* and *custom-3*
  - see also issues 457, 458 and 459