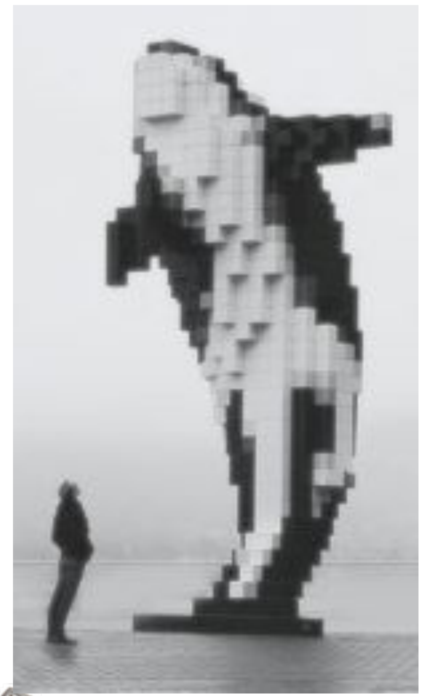
VectorBlox
embedded supercomputing

ORCA

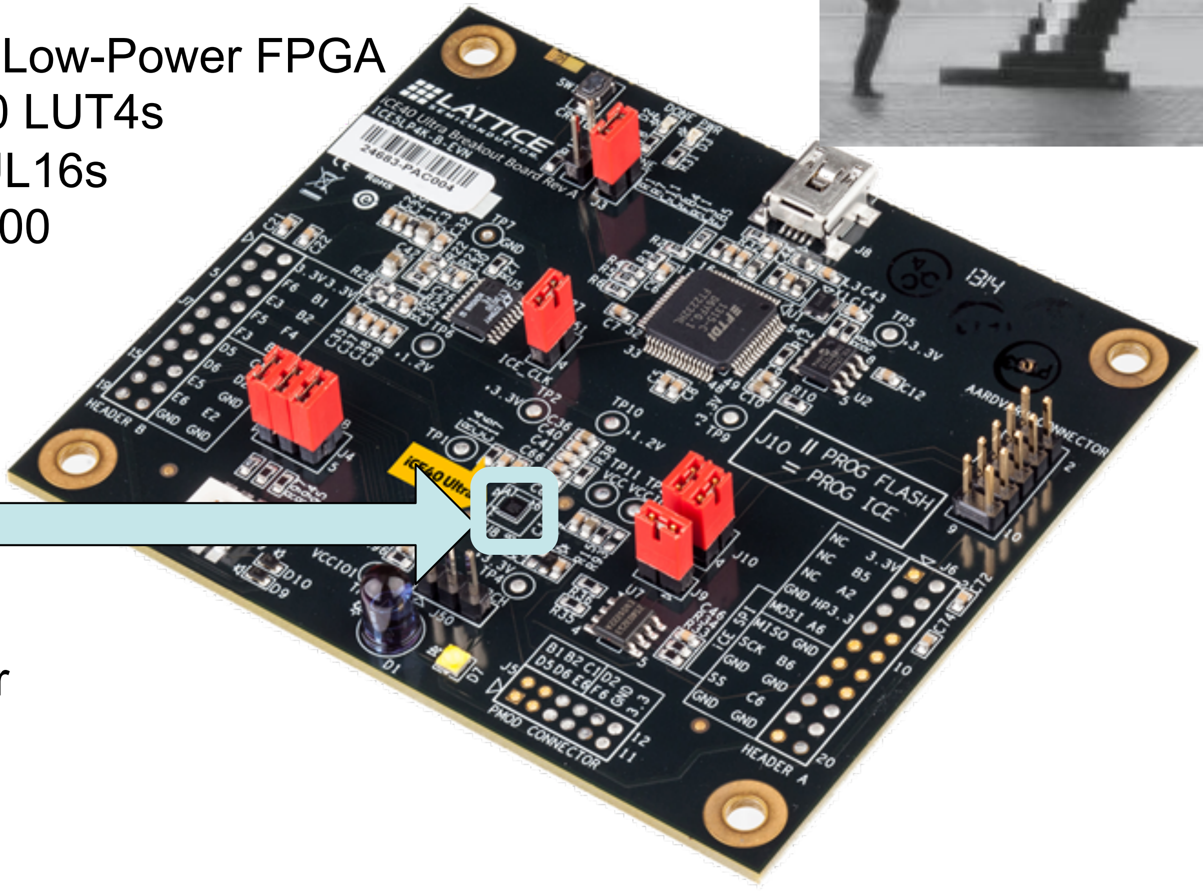
FPGA-Optimized RISC-V

VectorBlox
embedded supercomputing

ORCA
FPGA-Optimized RISC-V



Tiny, Low-Power FPGA
3,500 LUT4s
4 MUL16s
< $5.00



ISA: RV32IM
hw multiply, sw divider
< 2,000 LUTs
~ 20MHz

What is ORCA?

- Family of RISC-V implementations
  - Highly parameterized
  - Ideally suited for FPGAs
  - Portable across FPGA vendors
  - BSD license open source hardware

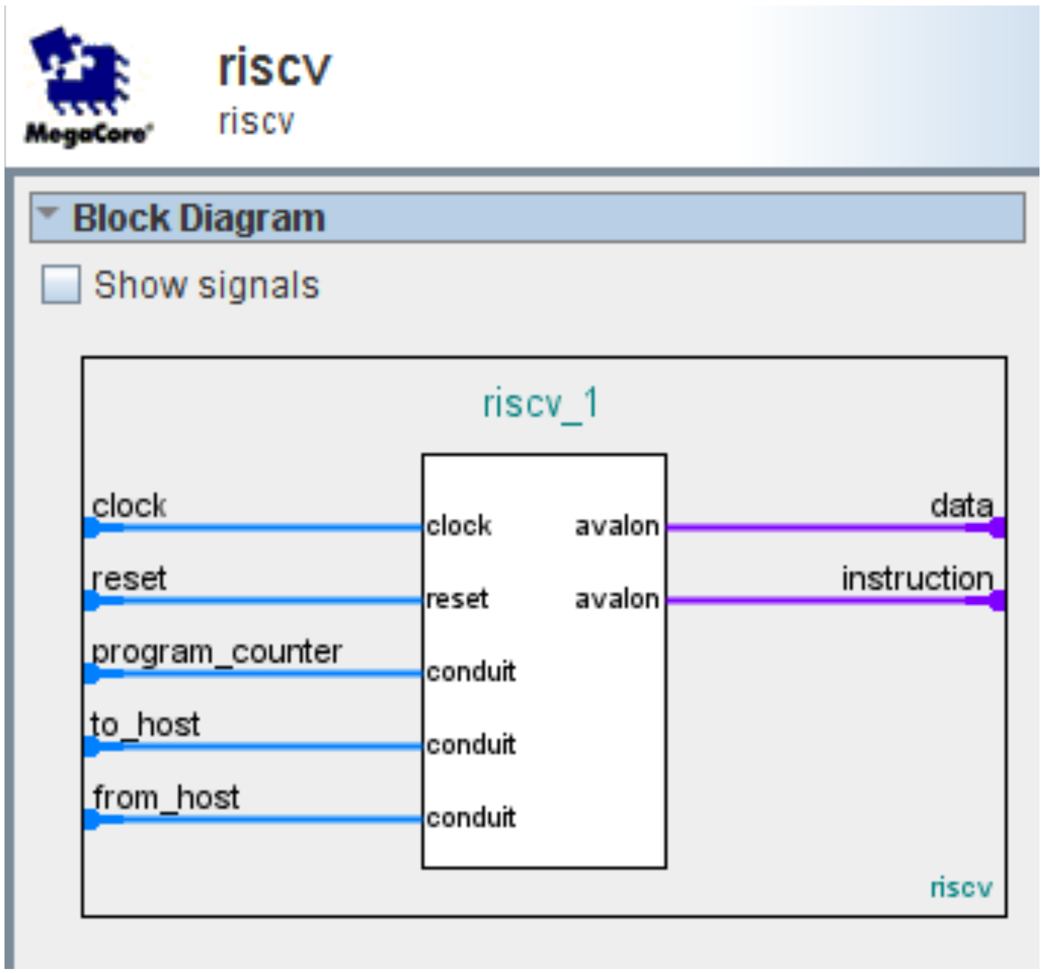
Why ORCA?

- Many reasons
  - Orcas travel in pods: family of many sizes
  - Orcas are native to Vancouver

- ORCA – many possible backronyms
  - ORCA RISC-V Computer Architecture
  - ORCA Reconfigurable CPU Architecture
  - Optimized RISC-V CPU Architecture

ORCA: Multiple FPGA Vendors

- Altera
  - Drop-in Qsys replacement for Nios II/f
  - Avalon I / D masters
- Lattice
  - Wishbone I / D masters
- Xilinx, Microsemi
  - Coming soon



ORCA RISC-V
RV32I on Different FPGAs

Area	2008 LUT4	1623 LUT4	541 ALMs
Fmax	22 MHz	109 MHz	244 MHz
DMIPS	n/a	79 MIPS	212 MIPS
DMIPS/MHz	n/a	0.73	0.87

ORCA vs Other RISC-V

	ORCA RV32IM	Z-scale RV32IM	PicoRV RV32I
Area	2353 LUT4 (Cyclone IV, 60nm)	2678 LUT4 (Spartan 6, 45nm)	2949 LUT4 (Cyclone IV, 60nm)
Fmax	125 MHz	33 MHz	127 MHz
DMIPS	122 MIPS	44 MIPS	39 MIPS
DMIPS/MHz	0.98 (measured)	1.35 (claimed)	0.31 (claimed)

ORCA RISC-V vs FPGA CPUs

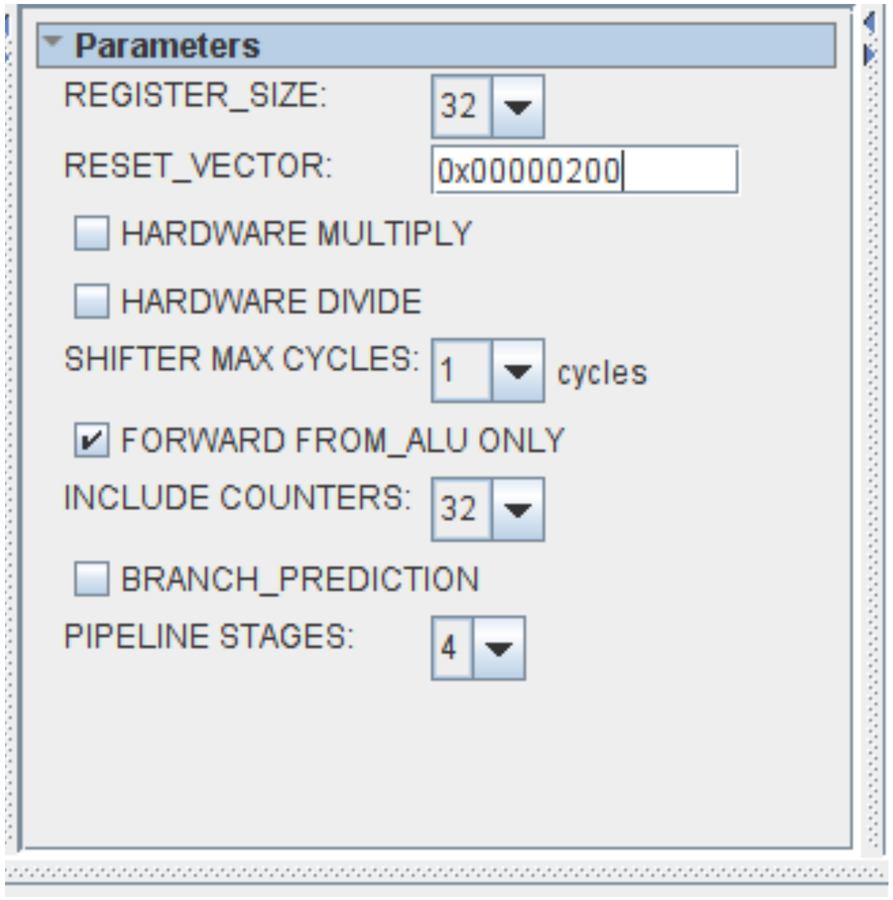
	ORCA RV32IM	**Altera Nios II/f**
Area	2353 LUT4 (Cyclone IV)	2678 LUT4 (Cyclone IV)
Fmax	125 MHz	140 MHz
DMIPS	122 MIPS	163 MIPS
DMIPS/MHz	0.98 (measured)	1.16 (claimed)

RISC-V: Architecture Space

- Width (3 choices)
  - 32, 64, 128 bits
- Instruction Set (9 binary options, 2^9 choices)
  - Minimum: I
  - Binary options: M, A, F, D (== G), Q, L, B, T, P
- Instruction Encoding (2 choices)
  - C
- Architecture Space 3 x 2^10 = 3072 possibilities

ORCA Implementation Space

- Logic design (12 choices)
  - Multiplier (sw, hw)
  - Divider (sw, hw)
  - Shifter (1-cycle, 8-cycles, 32-cycles)
- Counters (3 choices)
  - 0, 32, or 64 bits
- Pipelining (2 choices)
  - 4 or 5 stages
- Forwarding (2 choices)
  - ALU only
  - ALU + other units


Parameters

REGISTER_SIZE: 32

RESET_VECTOR: 0x00000200

[ ] HARDWARE MULTIPLY

[ ] HARDWARE DIVIDE

SHIFTER MAX CYCLES: 1 cycles

[✓] FORWARD FROM_ALU ONLY

INCLUDE COUNTERS: 32

[ ] BRANCH_PREDICTION

PIPELINE STAGES: 4

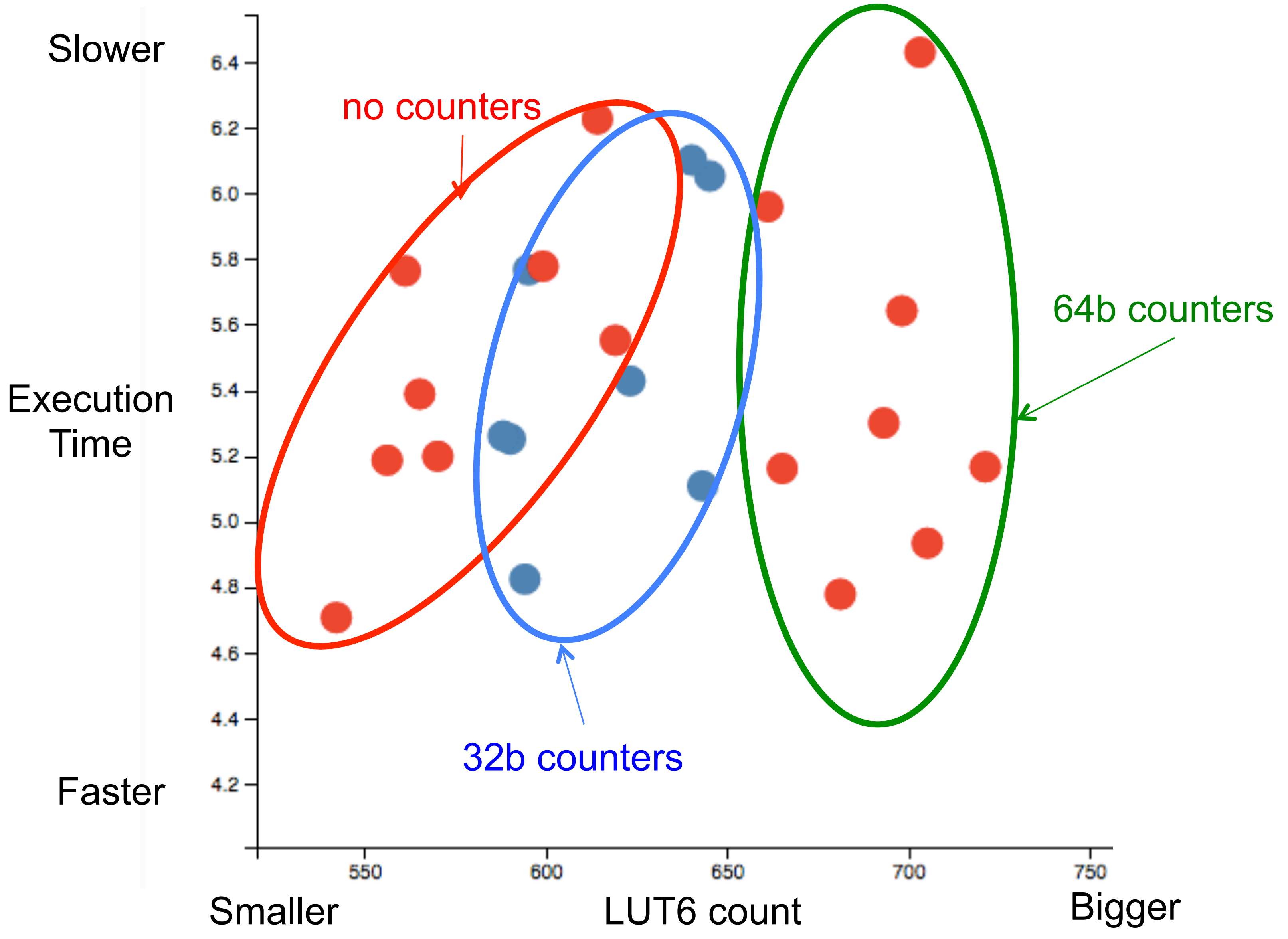
- Implementation space 12 x 3 x 2 x 2 = 144 possibilities
- Overall arch. x impl. = 3072 x 144 / 2 = 221,184 possibilities

Huge Design Space

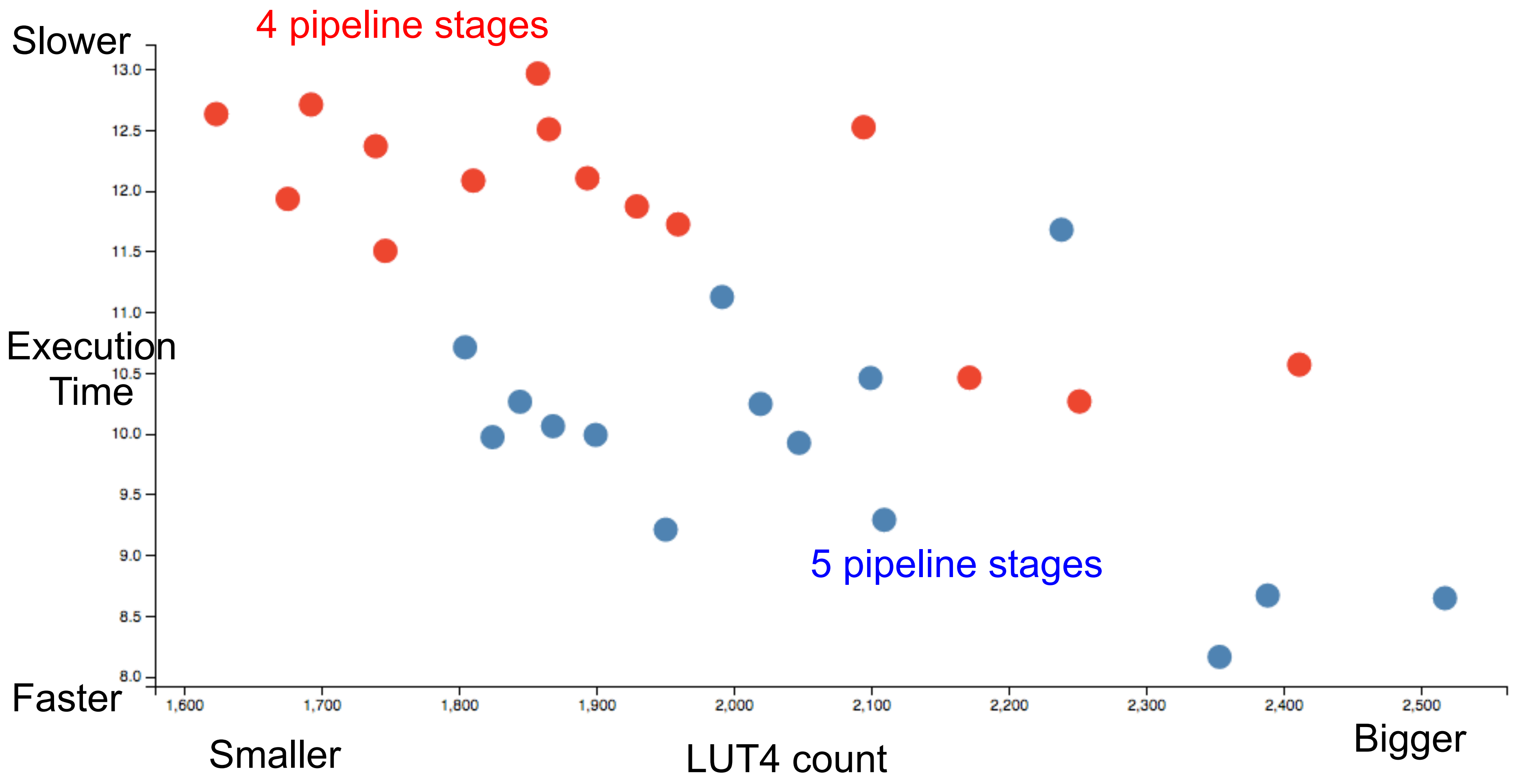
- Implementation on ASIC
  - Need to choose one design point in the architecture + implementation space
  - Benefit: user has no choice
  - Problem: compromise across many applications

- Implementation on FPGA
  - Can have fully parameterized design
  - User can choose best architecture + implementation according to application
  - Benefit: good performance, area
  - Problem: overwhelming design space

32b vs 64b Counters



4 vs 5 Pipeline Stages



FPGA ➔ ASIC
but
ASIC !➔ FPGA

good FPGA implementation

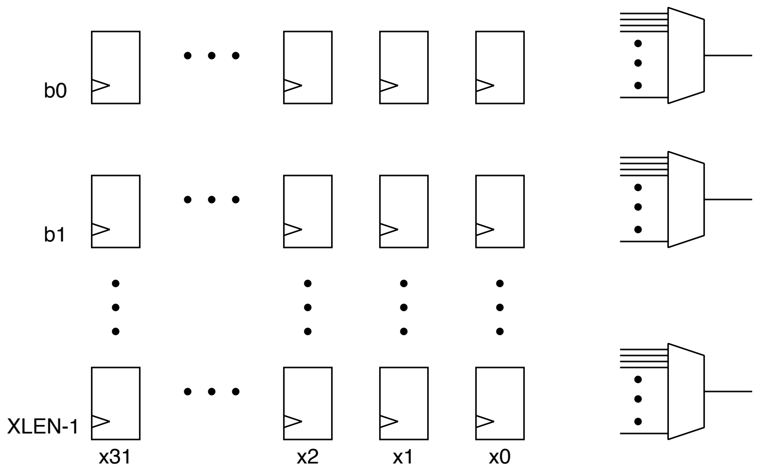
➔ often leads to good ASIC implementation

good ASIC implementation

➔ often leads to poor FPGA implementation

Register File

- Discrete FFs: inefficient



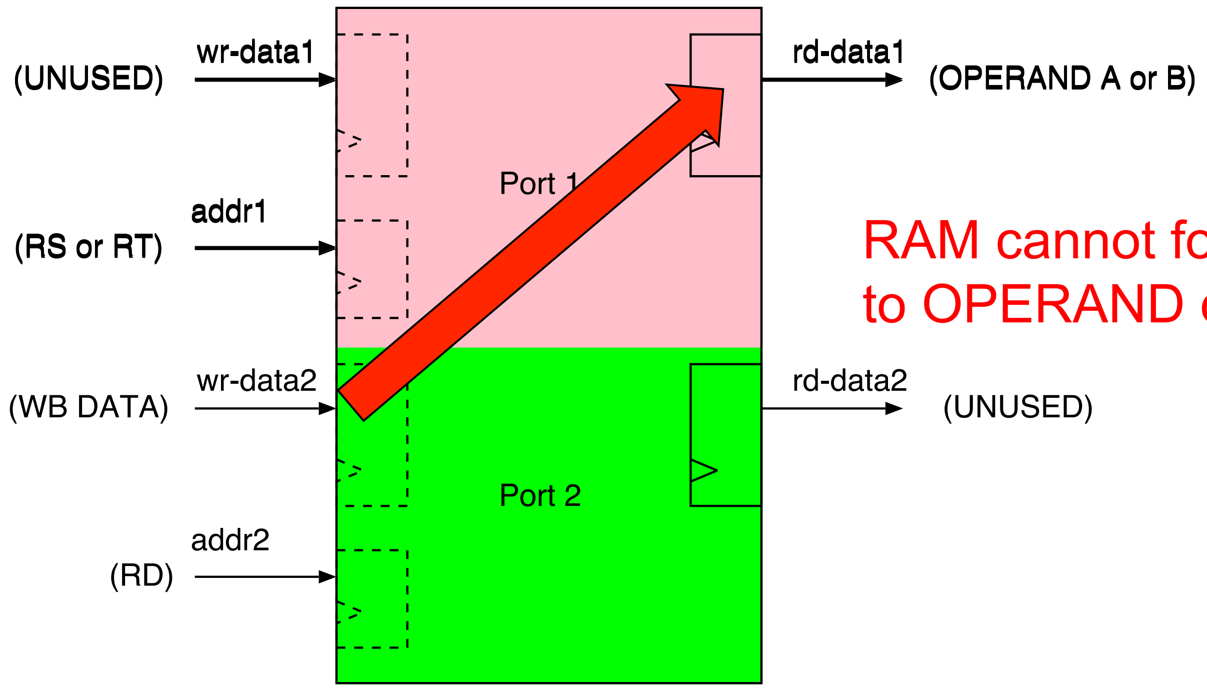
Cost of muxes (1b wide):

mux4:	2 LUT4	or 1 LUT6
mux16:	10 LUT4	or 5 LUT6
mux32:	11 LUT4	or 5.5 LUT6

  – 32 cpu registers x 32 b = 1024 FFs
  – 32 mux32 = 32 x 11 LUT4 = 352 LUT4s
- Note: muxes are costly, must avoid!!!

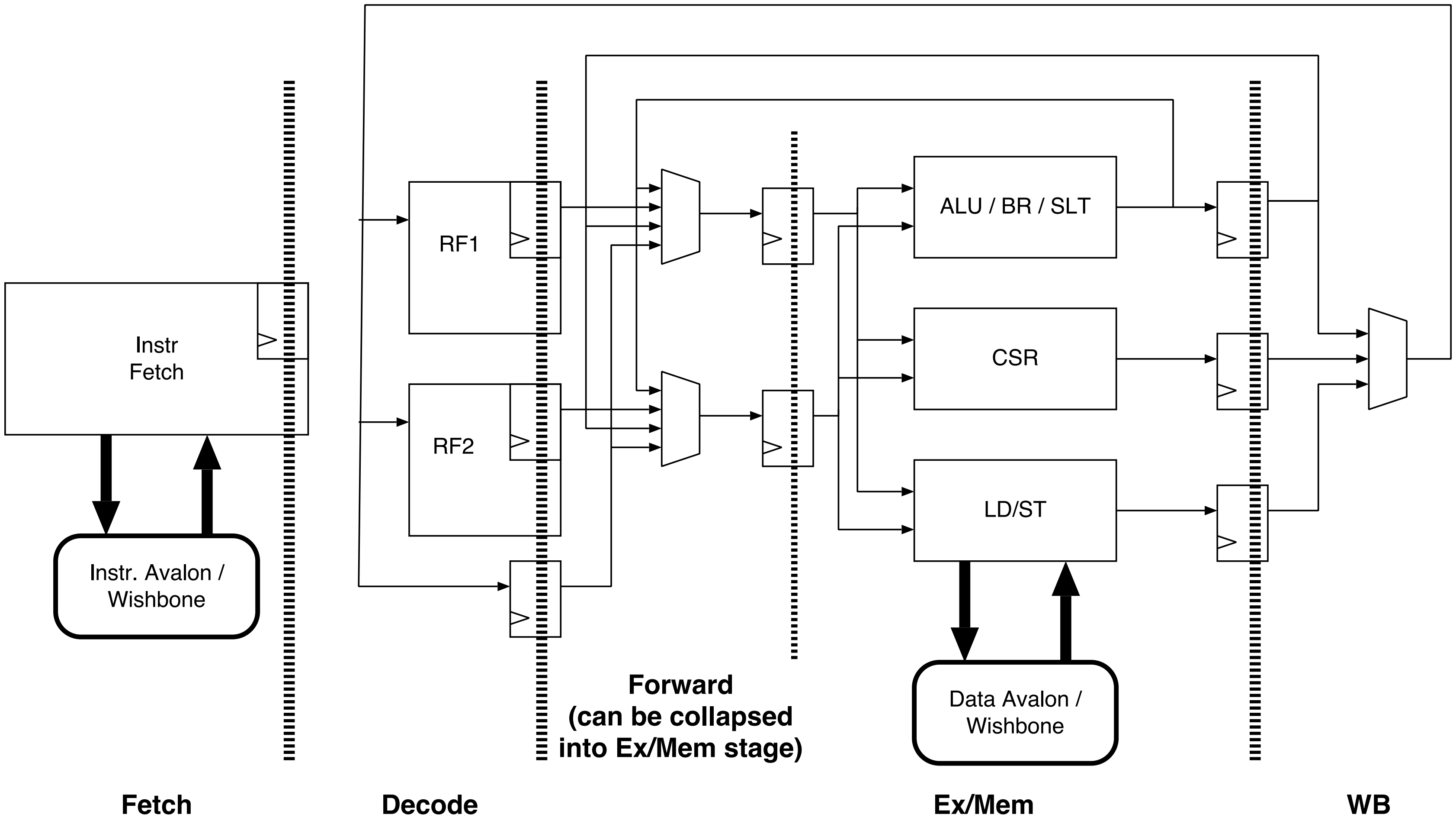
Register File Implications

- Block RAMs: dual ported, registered output
  - Use 1 RD, 1 WR port
  - Use data-out FFs as pipeline FFs
  - Needs external data-forwarding



RAM cannot forward WB data to OPERAND data internally

ORCA Datapath



Some FPGA Suggestions

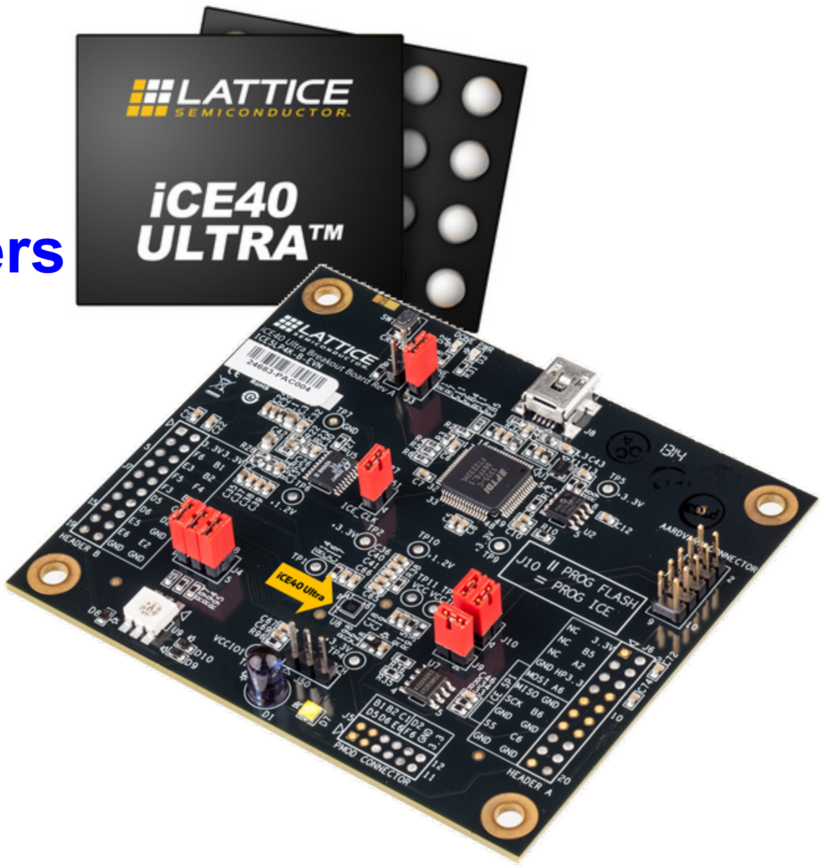
- RV32E spec
  - Reduced # registers saves nothing in FPGAs
  - Divide is expensive
- Software
  - Beware, shifts may be 1b/cycle (slow)
- Privileged Arch spec
  - Too many CSRs, 64b counters too big
    - Increases pressure on multiplexers
  - Suggest small / med / full versions
    - No “official” rules on what to include/exclude to reduce size
    - Eg, hypervisors not likely to run on FPGAs

Conclusions

- ORCA RISC-V family is free, portable, FPGA-optimized
  - FPGA and ASIC optimizations are different
    - FPGA architecture dictates certain design choices
  - Some RISC-V decisions are “unconsciously” aimed towards ASIC implementation
    - These do not lead to good FPGA implementations
    - But good FPGA choices lead to good ASICs

Free FPGA Hardware!

- **Today only: Lattice donating FPGA boards for RISC-V users**



VectorBlox embedded supercomputing

- ORCA RV32I system ~2000 LUTs

  http://www.github.com/VectorBlox/risc-v

  About 1500 LUTs available for user I/O (eg, UART)

LUNCHTIME

So Long, and Thanks for All the Fish !!