Ph.D Thesis Defense
Hardware Acceleration for Elementary Functions and RISC-V Processor

Presenter: Jing Chen
Supervisor: Prof. Xue (Steve) Liu

School of Computer Science
McGill University

24th, April, 2020

Outline

Motivation
What are elementary functions?

powers	x, x^2, x^3
reciprocal (2018)	$\frac{1}{x}$
root (2018)	$\sqrt{x}$
exponential	e^x
logarithm (2012)	$\log x$
trigonometric functions	$sin(x), cos(x), tan(x)$, etc.
inverse trigonometric functions	$arcsin(x), arccos(x), arctan(x)$, etc.
hyperbolic functions	$sinh(x), cosh(x)$, etc.

Motivation
Why are elementary functions important?

- Many scientific applications **heavily** rely on the evaluation of *floating-point* elementary functions
  - Digital and image signal processing applications [1]:
    - $1/x$, $\sqrt{x}$
  - Machine learning applications [2]:
    - $log(x)$
  - Web search and data analytics applications [3]:
    - $exp(x)$, $log(x)$ and $1/x$

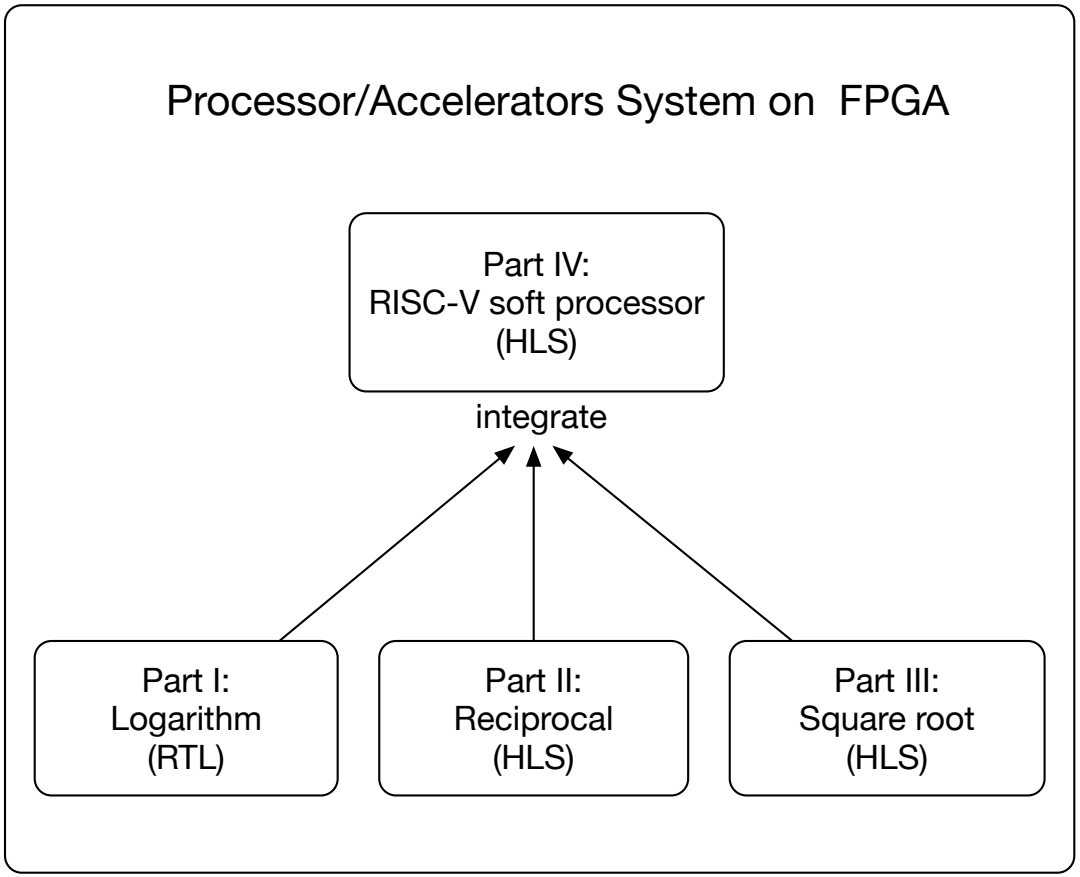
Background
Existing methods

- Series expansion:
  - Taylor series $\sum_{n=0}^{\infty} \frac{f^{(n)}(a)\,(x-a)^n}{n!}$
    - Pros: no memory usage
    - Cons: high precision, wide computation width, too many $\times$, $+$
- Lookup-table:
  - Function values are pre-computed and stored in a table
    - Pros: no run-time computation
    - Cons: high precision, large lookup-tables
- Lookup-table + Series expansion
  - Fewer $\times$, $+$
  - Smaller lookup-tables

Research Picture

Novelty

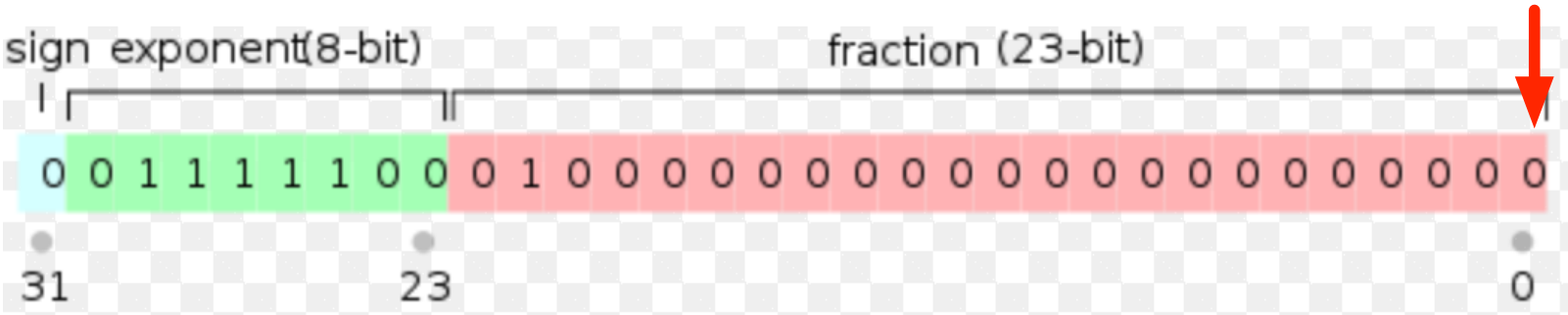
- Better performance (faster, smaller, more precise)
- Design method (more flexible, productive)



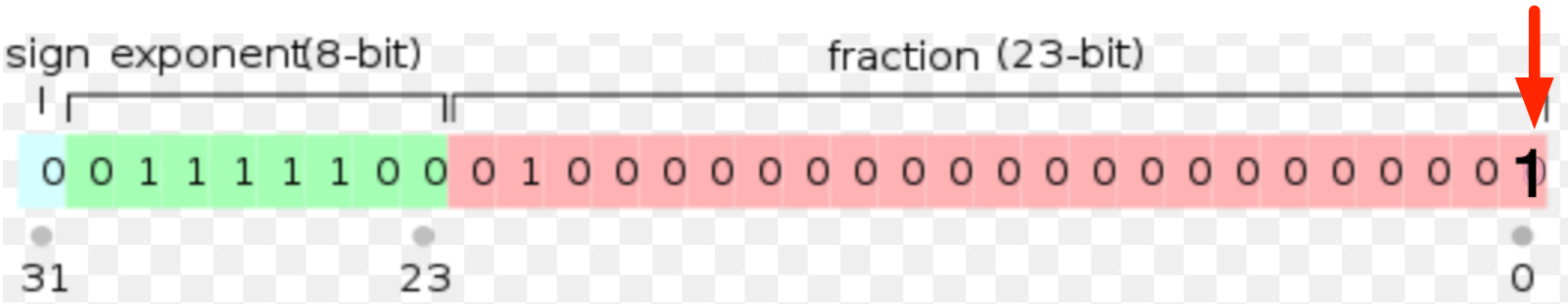
Part I: Logarithm Hardware Accelerator (McMaster, 2012)

- IEEE-754 single precision (32-bit) floating-point representation
- Algorithm: 7.8 KB LUT + degree-2 polynomial interpolation
- Architecture: pipelined accelerator, custom hardware
- Design method: **RTL**, VHDL, Quartus, Modelsim, Intel DE2
- Throughput: **2500 MHz**, $\approx$ **6000 MHz** deeper pipeline, 65 nm ASIC
- Precision: worst **3 ULP**, average **0.5 ULP**
- Area: $\approx$ **60,000** logic gates

What is 1 ULP accuracy?



0.156250000000000000000000



0.156250014901161193847656250

Part I: Logarithm Hardware Accelerator (McGill, 2017)

Enhanced and extended Master's research

- A new reduced-precision implementation
- Accuracy is re-measured using ULP rather than number of bits
- Throughput is re-calculated from 2900 MHz to 2500 MHz under 65nm ASIC
- Throughput can be tuned to 6000 MHz if using deeper pipeline architecture under 65nm ASIC
- A more thorough "Related Work"
- A more comprehensive "Performance Comparison"

Reciprocal & Square Root Hardware Accelerators Introduction

- Reciprocal: algorithm designed by myself
  - Trial subtraction (iterative)
  - Lookup-table ($\approx$1 KB) + degree-2 polynomial
- Square root: algorithm designed by myself
  - Newton's method (iterative)
  - Lookup-table ($\approx$1 KB) + degree-2 polynomial
- HLS tool: LegUp HLS, Prof. Anderson, University of Toronto
  - interprets C-software to Verilog code
  - generates hardware in pipeline architecture
- FPGA chip: Intel/Altera Cyclone V 45nm FPGA

Newton's Method

To evaluate $\sqrt{a}$, Newton's formula is:

$$x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n}) \tag{1}$$

- A good starting point x_0
- 3 divisions, 3 additions, and 3 shift operations
- 1 ULP precision required

Improved Lookup-Table Approach (compared to $\log x$)

- Smaller lookup-tables (6-16X smaller)
- Less computing cost for degree-2 polynomial
  - $ax^2 + bx + c \rightarrow x(ax + b) + c$
  - 2 multiplications, and 2 additions
  - all internal computation is fixed-point
- Narrower internal computing width (70-bit$\rightarrow$32-bit)
- Higher precision (1 ULP)

Table: Lookup-table for degree-2 polynomial

Entry	Content
E_0	a_0, b_0 , c_0
E_1	a_1, b_1, c_1
...	...
E_{1023}	a_{1023}, b_{1023}, c_{1023}

Part II: Reciprocal Hardware Accelerators
Performance Comparison

Table: Performance comparison for reciprocal accelerators

	Intel	Our work
Throughput (MHz)	310 (1.2X)	253.68
Pipeline depth	22	15 (68%)
ALMs	273	203 (74%)
Memory bits	30,208	10,112 (33%)
DSPs	6	2 (33%)
Accuracy (ULP)	1	1

- Intel IP cores: hand-designed RTL, highly-optimized
- Platform: Intel/Altera Cyclone V 45nm FPGA
- GNU math library, golden baseline
- Can achieve **6000 MHz** in 65nm if uses $\log x$ structure

Part III: Square Root Hardware Accelerators
Performance Comparison

Table: Performance comparison for square root accelerators

	Intel	Our work
Throughput (MHz)	310 (1.3X)	244.92
Pipeline depth	16	10 (62.5%)
ALMs	178	162 (91%)
Memory bits	15,872	10,624 (67%)
DSPs	3	2 (67%)
Accuracy (ULP)	1	1

- Intel IP cores: hand-designed RTL, highly-optimized
- Platform: Intel/Altera Cyclone V 45nm FPGA
- GNU math library, golden baseline
- Can achieve **6000 MHz** in 65nm if uses $\log x$ structure

Reduced-Precision, Reciprocal Hardware Accelerators

Table: Reduced-Precision reciprocal accelerators

	1 ULP	3 ULP
Throughput (MHz)	197.16	222.12 (1.1X)
Pipeline latency (ns)	45.6	31.5 (69%)
ALMs	150	115 (76%)
Memory bits	10,112	5,056 (50%)
DSPs	2	2

- 3 ULP: half-size lookup-table + degree-2 polynomial
- Easy to create because of HLS design methodology
  - would be time consuming to create in RTL design
- Platform: Intel/Altera Cyclone V 45nm FPGA
- GNU math library, golden baseline

Reciprocal and Square Root Hardware Accelerators Summary

Table: Performance analysis of reciprocal and square root accelerators

	Intel IP cores	Our work
Throughput	√	
Multipliers/adders width	√	
Table size		√
Circuit area		√
Flexibility		√
Productivity		√
Algorithm		√, universal

- More flexible: performance/area trade-offs, reduced-precision
- More productive: 1 year/me
- Better algorithm: small lookup-table + low-order polynomial

Part IV: RISC-V Soft-Processor
Motivation

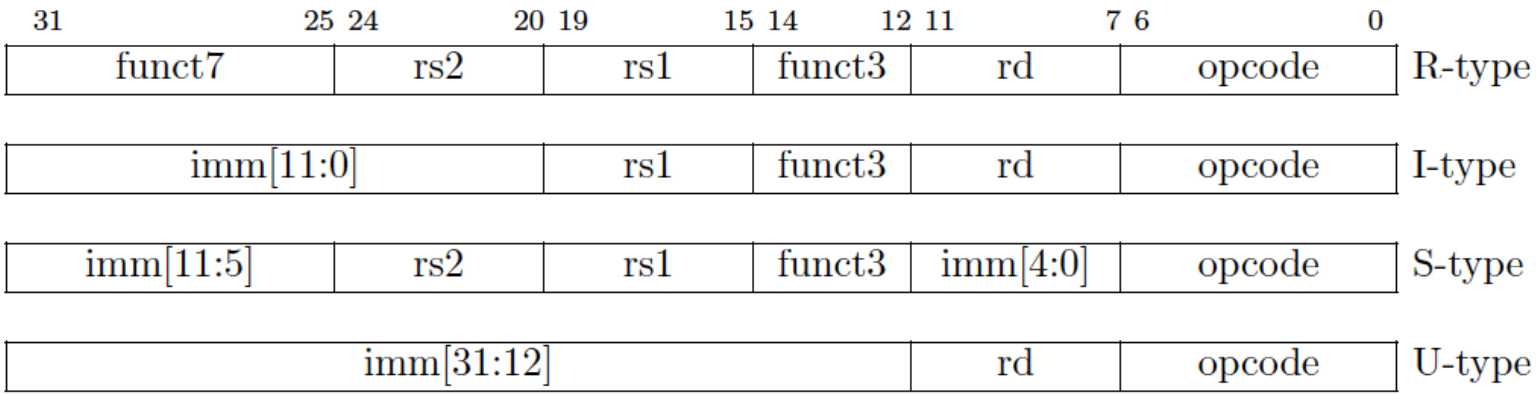
- Take advantage of the high performance accelerators ($\frac{1}{x}$, $\sqrt{x}$)
- Design a complete processor/accelerators system
- Easy to be extended for future research
- How to pick up a good **instruction set architecture (ISA)**?
  - open source
  - easy to understand/implement
  - active/widely supported
  - good reputation
  - sufficient software/hardware tools

Part IV: RISC-V Soft-Processor
Advantages of RISC-V

- RISC-V foundation, open source, non-profit
- More than 500 members globally [4]
  - IBM, Google, Nvidia, Xilinx, HP, Qualcomm, SIEMENS, NOKIA, HITACHI, SONY, HUAWEI...
- Dr.David Patterson **(2017 Turing Award Winner)**
- UC Berkeley team, began in 2010
- Early funding, Defense Advanced Research Projects Agency
- Efficient in all domains (server, mobile, embedded)
- Efficient in all implementation technology (ASIC, FPGA)
- Free simulator, OS, compiler, library...

Part IV: RISC-V Soft-Processor
Features of RISC-V ISA

1. Configurable/Extensive (growing rapidly!)
2. ISA= Base + Extensions
   - Base instruction sets
     - **32-bit, 47 instructions**
     - 64-bit, 62 instructions...
   - Extension instruction sets [4]
     - M: integer multiplication, division
     - F: single-precision (32-bit)
     - D: double-precision (64-bit)
     - Q: quad-precision (128-bit)...



Part IV: RISC-V Soft-Processor
Our Work

- RISC-V 32-bit processor: 39 user instructions
- Design language, tools: C-software, LegUp HLS, Intel FPGA
- Productivity: 695 lines of C, 1 year/(me, Prof. Anderson)
- Flexibility: ease of change, various performance/area trade-offs
- Testing: bubble sort, fibonacci-sequence generator, factorial...
  - GCC toolflow, C-software $\rightarrow$ RISC-V machine code
- Architecture: (1) multi-cycle, and (2) pipeline

Novelty

- Significant productivity boost
- Equivalent/even better performance
- HLS works well for processor implementation!

Part IV: RISC-V Soft-Processor Performance Comparison

Table: Performance comparison of pipelined RISC-V soft-processors

	ORCA [5]	Our work
Maximum Frequency (MHz)	122	125.6 (1.02X)
Circuit Area (ALMs)	992	682 (69%)
Instructions/Cycle	0.6-0.7 (3-3.5X)	0.2
Line of Code (Productivity)	4,400	695 (16%)
Language (Flexibility)	VHDL	C

- ORCA: open source, UBC startup, Microsemi, frequently cited
- Platform: Intel/Altera Cyclone V 45nm FPGA

Conclusions

- Complete processor/accelerators system on FPGA
  - 32-bit RISC-V soft processor
  - elementary functions accelerators
- Elementary functions accelerators
  - faster, smaller, more precise
- HLS (C-software), FPGA
  - productivity, flexibility
  - ease of change, future computing

Future Work

- Power consumption
- An entire library of elementary functions, universal method
- From multi-cycle to advanced pipeline RISC-V processor
- Integrate the accelerators by extending the RISC-V ISA
- Highly-optimized processor/accelerators system from C-software!

Thank You!

Reference

- L. Moroz, V. Samotyy, and O. Horyachyy, "An effective floating-point reciprocal," in *The 4th IEEE International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems*. Lviv, Ukraine, 2018, pp. 280–285.
- N. Alachiotis and A. Stamatakis, "Efficient floating-point logarithm unit for FPGAs," in *IEEE Int'l Symp. on Parallel & Distributed Processing Workshops and PhD Forum*, 2010, pp. 1–8.
- M. Langhammer and B. Pasca, "Design and implementation of an embedded FPGA floating point DSP block," in *IEEE ARITH*, 2015, pp. 26–33.
- "The RISC-V Instruction Set Manual. [Online]. Available:," https://riscv.org/specifications/, 2019.
- "OCRA: RISC-V by VectorBlox. [Online]. Available:," https://github.com/VectorBlox/orca, 2019.