Vendor Agnostic, High Performance, Double Precision Floating Point Division for FPGAs

Presented by Xin Fang

Advisor:
Professor Miriam Leeser

ECE Department
Northeastern University

Outline

- Background
- Algorithm Description
- Components of Divider
- Results & Performance
- Conclusions and Future work

Background

- Floating Point Format
- VFloat Library
- Three Main Algorithms

Floating Point Representation

- IEEE Standard for Floating-Point Arithmetic

(*IEEE 754-2008*)

Format	Sign Bit (s)	Exponent Bits (e)	Mantissa Bits (c)
Single Precision	1	8	23
Double Precision	1	11	52

b is the base, s is the sign bit, e is the exponent, c is the mantissa. The value of floating point number is

$$(-1)^s * 1.c * b^{e-bias}$$

Floating Point Format(Cont)

$$(-1)^s * 1.c * b^{e-bias}$$

- Mantissa Representation
- Exponent Bias

Format	Exponent in IEEE754	Exponent Bias	Real Exponent
Single Precision	1 to 254	127	-126 to 127
Double Precision	1 to 2046	1023	-1022 to 1023

VFloat Library

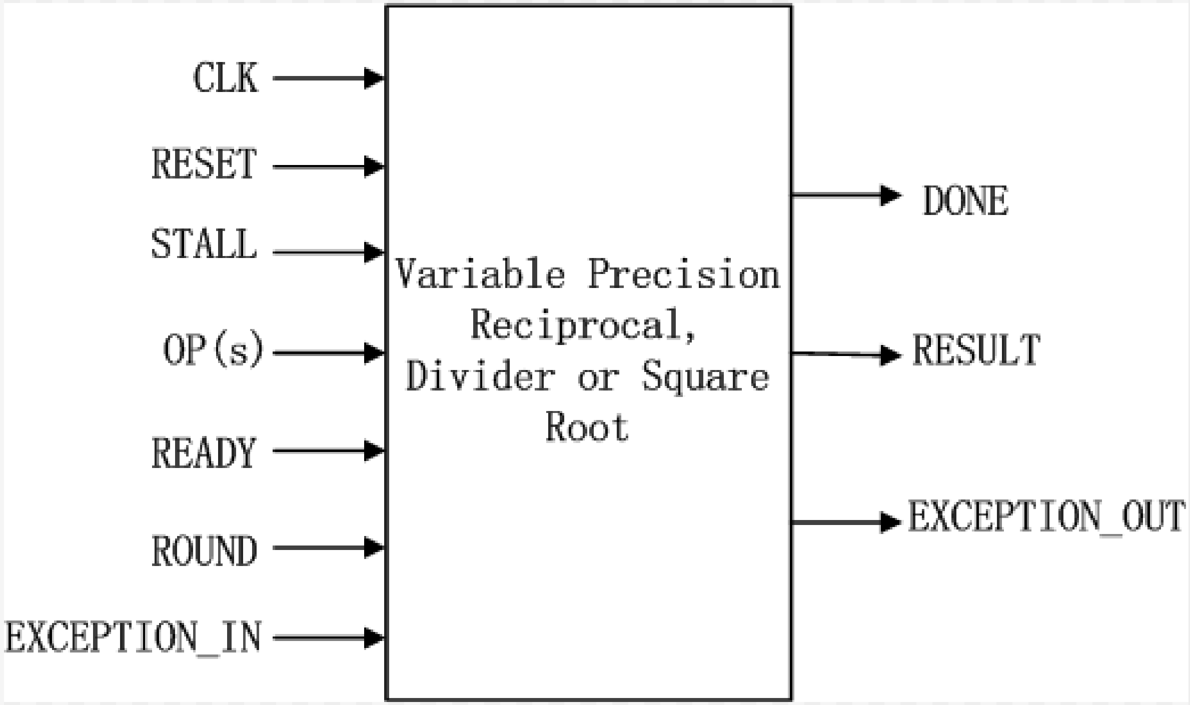
- Vfloat*: library of variable precision floating point units written in VHDL, targeting Altera and Xilinx FPGAs
- Components include floating point arithmetic (addition, subtraction, multiplication, reciprocal, division, square root, accumulation) and format conversion (fix2float).
- Operand can be variable precision floating point number

  Any bitwidth exponent or mantissa is supported.

*http://www.coe.neu.edu/Research/rcl/projects/floatingpoint/index.html

VFloat Library(Cont)

Blackbox for Variable Precision Floating Point Operation:



Components of VFloat can be easily fit into a large project because of the ports Reset, Stall, Ready, Exception_in, Exception_out and Done.

Three Main Approaches

- For Division:

- 1. Digit Recurrence Method
- 2. Iterative Method
- 3. Table-based Method

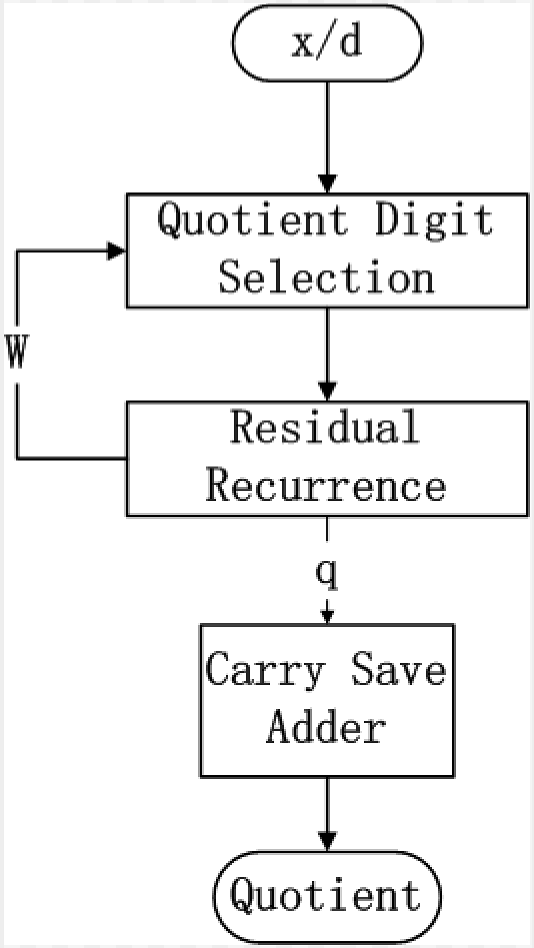
Digit Recurrence Division

- Example: SRT Algorithm
  - Step1: Quotient Digit Selection

$$q_{j+1} = \begin{cases} 1 & 2 * W_j >= 1/2; \\ 0 & -1/2 =< 2 * W_j < 1/2; \\ -1 & 2 * W_j < -1/2; \end{cases}$$

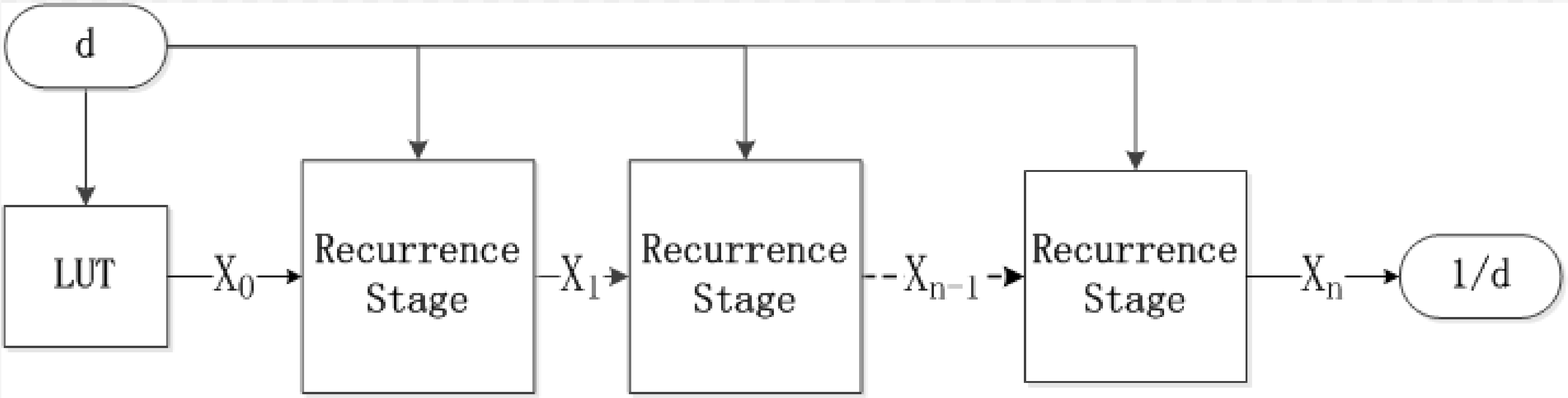
  - Step 2: Residual Recurrence

$$w[j+1] = r * w[j] - d * q[j+1]$$



Iterative Division

- Example: Newton Raphson Algorithm
  - Step 1: find an approximation of 1/d from LUT
  - Step 2: Iteration: $X_{(i+1)} = X_i \times (2 - d \times X_i)$



Newton Raphson Algorithm Diagram

Table-based Division

- Table-based algorithm* that uses small look-up tables and multipliers
  - Based on Taylor series, but more sophisticated
- Gets good tradeoff between clock cycle latency, maximum frequency and resource utilization by implementing this table-based algorithm
- Double Precision Division in this presentation is based on this algorithm

* Ercegovac, Milos D., et al. "Reciprocation, square root, inverse square root, and some elementary functions using small multipliers." *Computers, IEEE Transactions on* 49.7 (2000): 628-637.

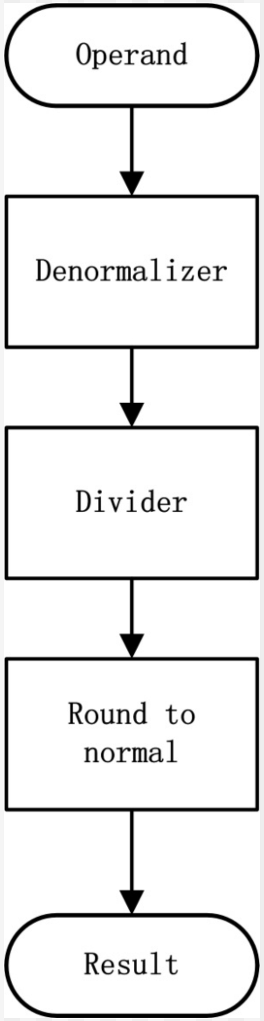
Outline

- Background
- Algorithm Description
- Components of Divider
- Results & Performance
- Conclusions and Future work

Algorithm Description

- Algorithm Overview
- Mantissa Computation:
  1. Reduction Step
  2. Evaluation Step
  3. Post-processing Step

Algorithm Overview



Top Components:

1. Denormalizer
2. Divider
3. Round to normal

Algorithm Overview(Cont)

- Three parts: sign bit, exponent and mantissa

- **Sign bit**: XOR Logic
- **Exponent:**

  Subtract the exponent of divisor from that of dividend
- **Mantissa:** where Table-based algorithm applies

Mantissa Computation

- Three steps to compute Mantissa:
  - 1. Reduction Step (To compute M)
  - 2. Evaluation Step (To compute B)
  - 3. Post-processing Step (To compute Result)

1. Reduction Step

After the denormalizer, inputs (X, Y) are the significand of the floating point number with format 1.c

each of them has m bits.

For double precision format, m = 53.

Let K = floor((m+2)/4)+1

Y(K) represents the truncation of Y to the first K bits

Let R = 1/ Y(K);

M = R

2. Evaluation Step

- Let B = f(A) = 1/(1+A)

  Where A = (Y*R)-1, $(-2^{-k} < A < 2^{-k})$

  Let $z = 2^{-k}$, then $A = A_2 z^2 + A_3 z^3 + A_4 z^4 + \ldots$

  A :

A1	**A2**	**A3**	**A4**

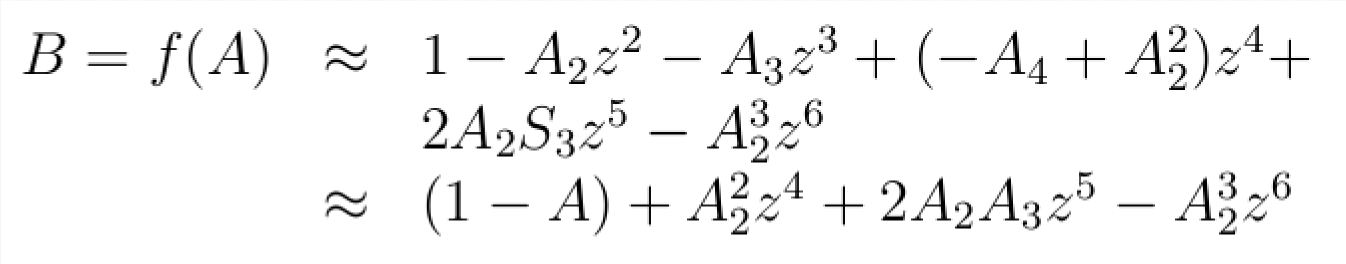
Using Taylor series expansion,

$$\begin{aligned} B = f(A) &= C_0 + C_1 A + C_2 A^2 + C_3 A^3 + C_4 A^4 + \cdots \\ &\approx C_0 + C_1(A_2 z^2 + A_3 z^3 + A_4 z^4) \\ &\quad + C_2(A_2 z^2 + A_3 z^3 + A_4 z^4) \\ &\quad + C_3(A_2 z^2 + A_3 z^3 + A_4 z^4)^3 \\ &\quad + C_4(A_2 z^2 + A_3 z^3 + A_4 z^4)^4 \\ &\approx C_0 + C_1 A + C_2 A_2^2 z^4 + 2C_2 A_2 A_3 z^5 + C_3 A_2^3 z^6 \end{aligned}$$

C_i = 1 when i is even, C_i = 0 when i is odd.

Evaluation Step(Cont)

- Simplifying the equation:



$$\begin{aligned} B = f(A) &\approx 1 - A_2 z^2 - A_3 z^3 + (-A_4 + A_2^2) z^4 + \\ &\quad 2A_2 S_3 z^5 - A_2^3 z^6 \\ &\approx (1 - A) + A_2^2 z^4 + 2A_2 A_3 z^5 - A_2^3 z^6 \end{aligned}$$

3. Post-processing Step

- The result of reciprocal, divider or square root is given by M * B

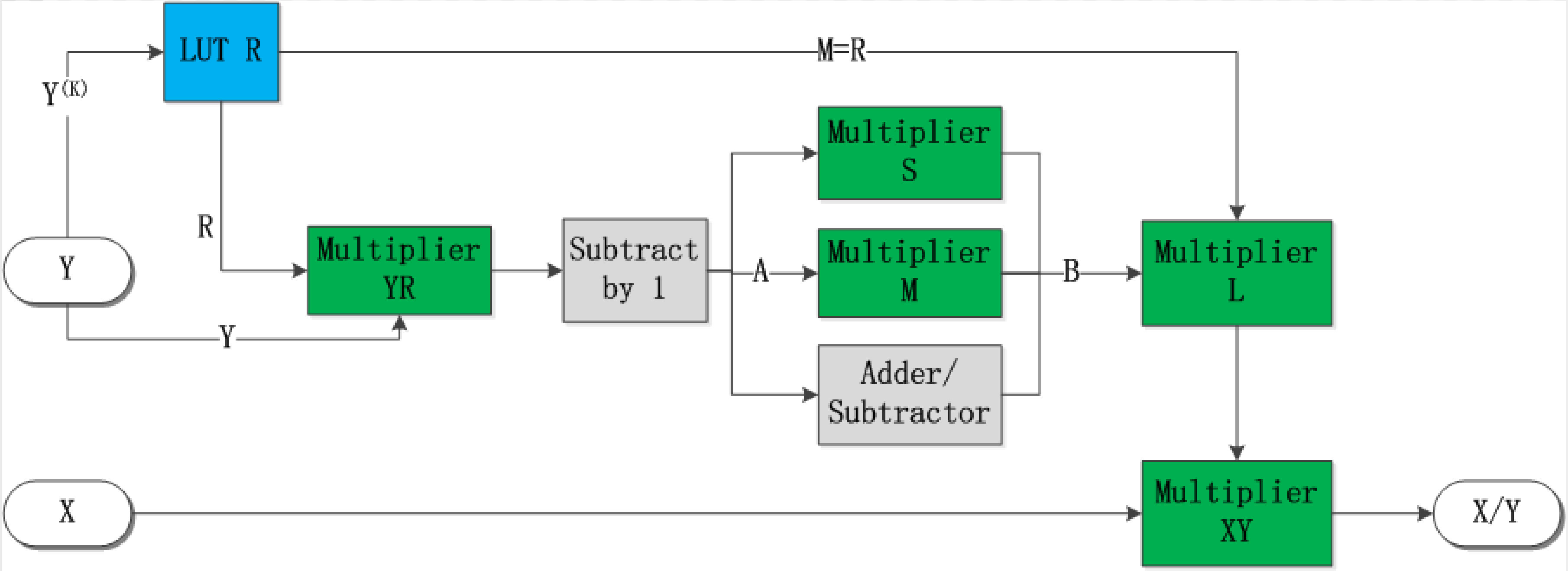
  M is from the Reduction step

  B is from the Evaluation step

Outline

- Background
- Algorithm Description
- Components of Divider
- Results & Performance
- Conclusions and Future work

Components of Divider



Outline

- Background
- Algorithm Description
- Components of Divider
- Results & Performance
- Conclusions and Future work

Results and Performance

- FPGA Device:

1. Stratix V device
2. Virtex 6 device

- Software IDE:

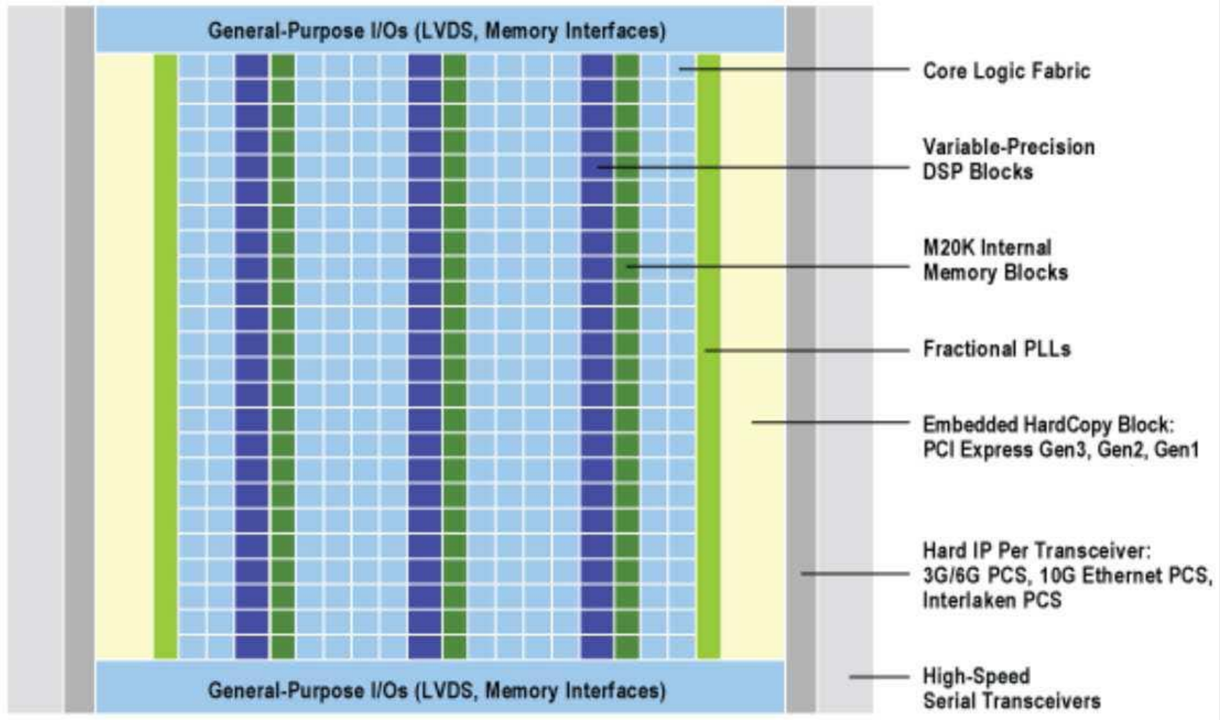
1. Altera Quartus II 13.0
2. Xilinx ISE Design Suite 13.4

Altera Stratix V

- Optimized for high performance, high bandwidth applications. (28-ns)
- The Stratix V device family contains GT, GX, GS and E sub-families.
- 5SGXB6 for synthesis.

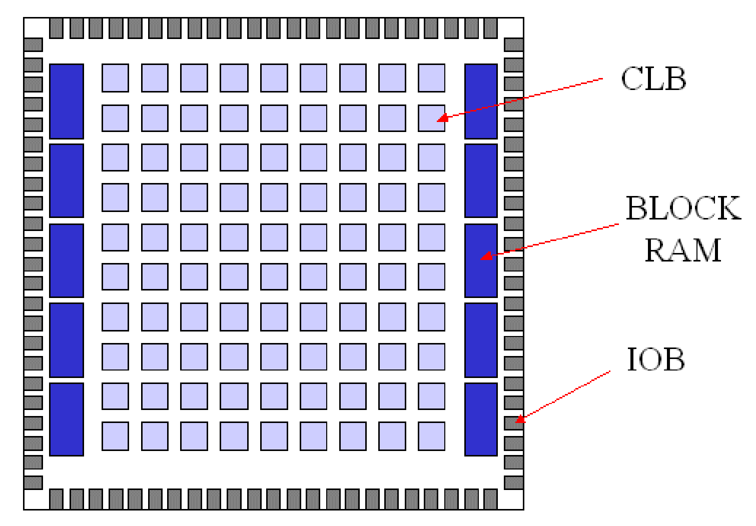
Altera Stratix V (5SGXB6)

- 597K Logic Elements
- 902K Register
- 2660 M20K (52 Mb)
- 798 18*18 Mul
- 399 27*27 Mul


General-Purpose I/Os (LVDS, Memory Interfaces)
Core Logic Fabric
Variable-Precision DSP Blocks
M20K Internal Memory Blocks
Fractional PLLs
Embedded HardCopy Block: PCI Express Gen3, Gen2, Gen1
Hard IP Per Transceiver: 3G/6G PCS, 10G Ethernet PCS, Interlaken PCS
High-Speed Serial Transceivers
General-Purpose I/Os (LVDS, Memory Interfaces)

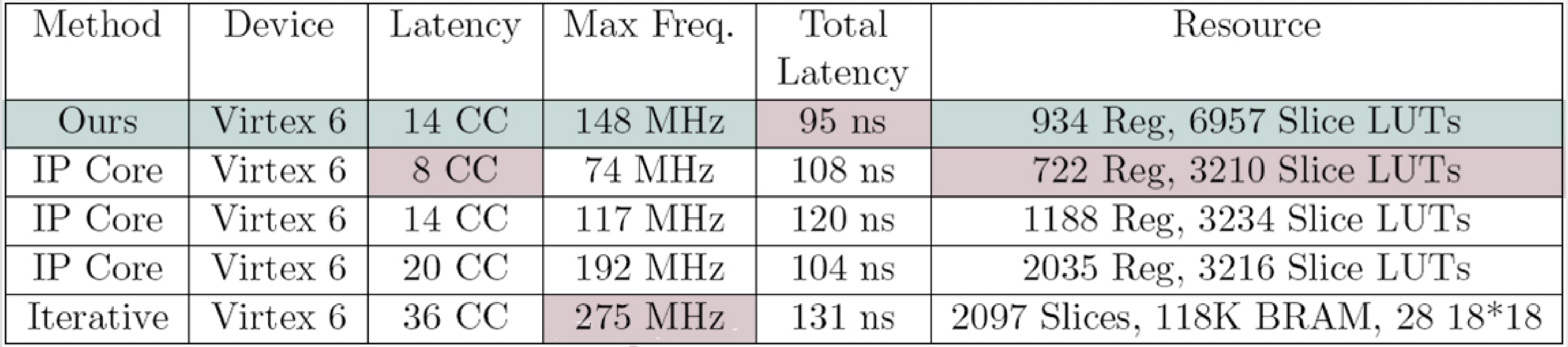
Xilinx Virtex 6

- High performance programmable silicon
- Three sub-family: LXT, SXT and HXT.
- XC6VLX75T for synthesis.
  - 74,496 Logic Cells
  - 11,640 Slices
  - 1,045 Kb Distributed RAM
  - 288 DSP48E1 Slices
  - 156 36Kb Block RAM



Results & Performance (Double Precision Division)

- For Xilinx:

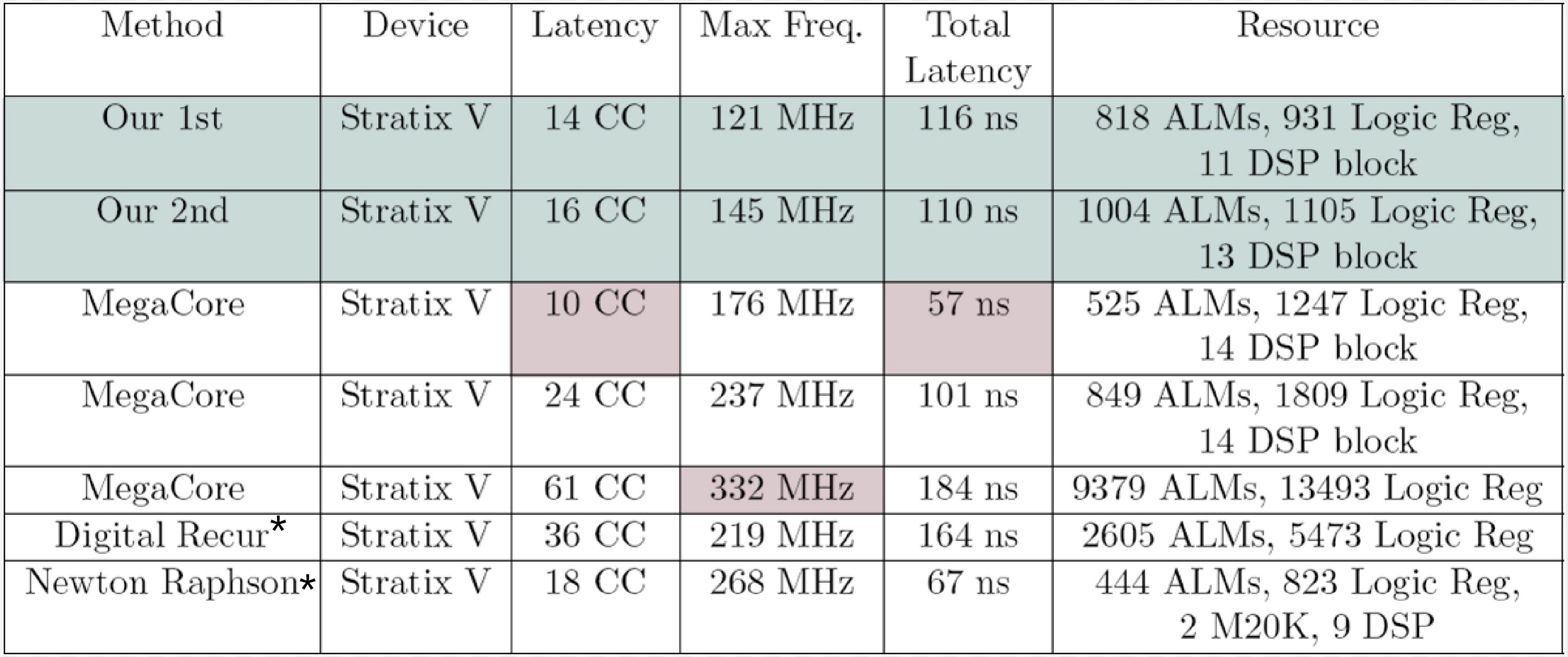


Method	Device	Latency	Max Freq.	Total Latency	Resource
Ours	Virtex 6	14 CC	148 MHz	95 ns	934 Reg, 6957 Slice LUTs
IP Core	Virtex 6	8 CC	74 MHz	108 ns	722 Reg, 3210 Slice LUTs
IP Core	Virtex 6	14 CC	117 MHz	120 ns	1188 Reg, 3234 Slice LUTs
IP Core	Virtex 6	20 CC	192 MHz	104 ns	2035 Reg, 3216 Slice LUTs
Iterative	Virtex 6	36 CC	275 MHz	131 ns	2097 Slices, 118K BRAM, 28 18*18

* Result is from Pasca, Bogdan. "Correctly rounded floating-point division for DSP-enabled FPGAs." *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*. IEEE, 2012.

Results & Performance (Double Precision Division)

- For Altera:



Method	Device	Latency	Max Freq.	Total Latency	Resource
Our 1st	Stratix V	14 CC	121 MHz	116 ns	818 ALMs, 931 Logic Reg, 11 DSP block
Our 2nd	Stratix V	16 CC	145 MHz	110 ns	1004 ALMs, 1105 Logic Reg, 13 DSP block
MegaCore	Stratix V	10 CC	176 MHz	57 ns	525 ALMs, 1247 Logic Reg, 14 DSP block
MegaCore	Stratix V	24 CC	237 MHz	101 ns	849 ALMs, 1809 Logic Reg, 14 DSP block
MegaCore	Stratix V	61 CC	332 MHz	184 ns	9379 ALMs, 13493 Logic Reg
Digital Recur*	Stratix V	36 CC	219 MHz	164 ns	2605 ALMs, 5473 Logic Reg
Newton Raphson*	Stratix V	18 CC	268 MHz	67 ns	444 ALMs, 823 Logic Reg, 2 M20K, 9 DSP

*Result is from Pasca, Bogdan. "Correctly rounded floating-point division for DSP-enabled FPGAs." *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*. IEEE, 2012.

Outline

- Background
- Algorithm Description
- Components of Divider
- Results & Performance
- Conclusions and Future work

Conclusions

- Double Precision Floating Point Division component using table-based algorithm
- Flexible: Easy to adjust pipeline parameters
- Adaptable: Cross-platform portability
- Good tradeoff between clock cycle latency, maximum frequency and resource utilization

Future Work

- Try different pipelines to find better tradeoffs
- Add more components to the library
- Apply to applications

Thank you!

- Xin Fang fang.xi@husky.neu.edu
  ECE Department, Northeastern University
  Miriam Leeser mel@coe.neu.edu
  ECE Department, Northeastern University

- VFloat website:
  http://www.coe.neu.edu/Research/rcl/projects/floatingpoint/index.html