Variable Precision Floating Point Division and Square Root

Albert Conti
Xiaojun Wang
Dr. Miriam Leeser
Rapid Prototyping Laboratory
Northeastern University, Boston MA
http://www.ece.neu.edu/groups/rpl/

Outline

- Project overview
- Library hardware modules
- Floating point divider and square root
- K-means clustering application for multispectral satellite images using the floating point library
- Conclusions and future work

Variable Precision Floating Point Library

- A library of fully pipelined and parameterized floating point modules
- Implementations well suited for state of the art FPGAs
  - Xilinx Virtex II FPGAs and Altera Stratix devices
  - Embedded Multipliers and Block RAM
- Signal/image processing algorithms accelerated using this library

Why Floating Point (FP) ?

Fixed Point

- Limited range
- Number of bits grows for more accurate results
- Easy to implement in hardware

Floating Point

- Dynamic range
- Accurate results
- More complex and higher cost to implement in hardware

Floating Point Representation

Sign	Biased exponent	Mantissa m=1.f (the 1 is hidden)
+/-	e+bias	f

32-bits:	8 bits, bias=127	23+1 bits, IEEE single-precision format
64-bits:	11 bits, bias=1023	52+1 bits, IEEE double-precision format

$$(-1)^{s} * 1.f * 2^{e\text{-}BIAS}$$

Why Parameterized FP ?

- Minimize the bitwidth of each signal in the datapath
  - Make more parallel implementations possible
  - Reduce the power dissipation
- Further acceleration
  - Custom datapaths built in reconfigurable hardware using either fixed-point or floating point arithmetic
  - Hybrid representations supported through fixed-to-float and float-to-fixed conversions

Outline

- Project overview
- Library hardware modules
- Floating point divider and square root
- K-means clustering application for multispectral satellite images using the floating point library
- Conclusions and future work

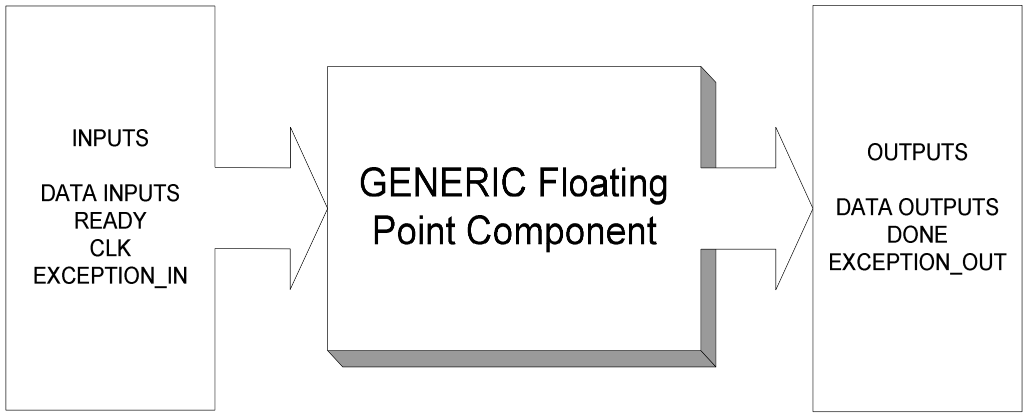
Parameterized FP Modules

- Arithmetic operation
  - **fp_add** : floating point addition
  - **fp_sub** : floating point subtraction
  - **fp_mul** : floating point multiplication
  - **fp_div** : floating point division
  - **fp_sqrt** : floating point square root
- Format control
  - **denorm** : introducing implied integer digit
  - **rnd_norm** : rounding and normalizing
- Format conversion
  - **fix2float** : converting from fixed point to floating point
  - **float2fix** : converting from floating point to fixed point

What Makes Our Library Unique ?

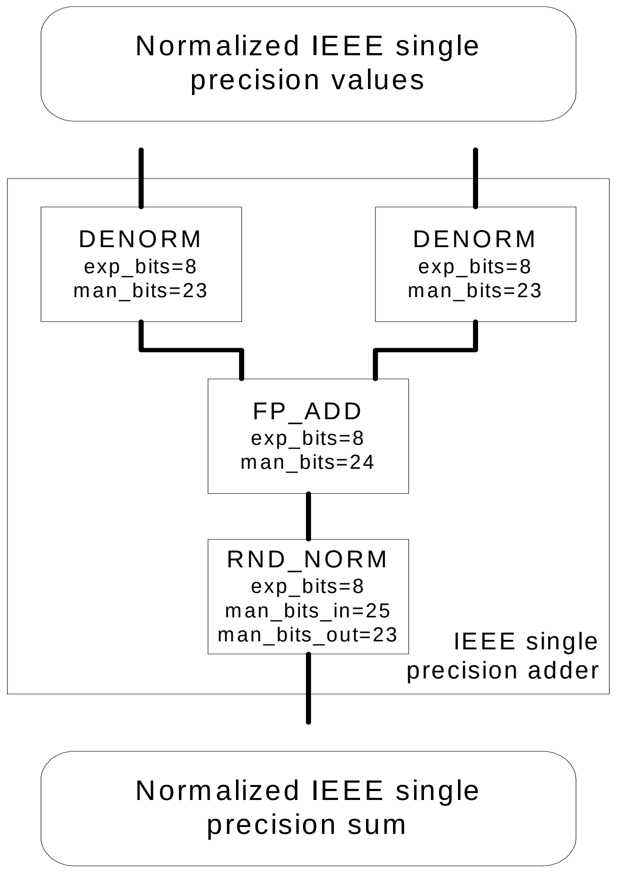
- A superset of all floating point formats
  - including IEEE standard format
- Parameterized for variable precision arithmetic
  - Support custom floating point datapaths
  - Support hybrid fixed and floating point implementations
- Support fully pipelining
  - Synchronization signals
- Complete
  - Separate normalization
  - Rounding (“round to zero” and “round to nearest”)
  - Some error handling

Generic Library Component



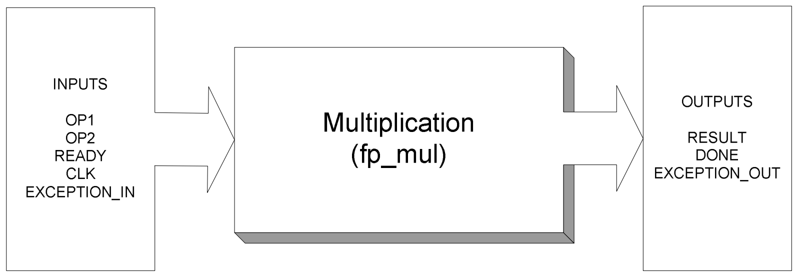
- Synchronization signals for pipelining
  - READY and DONE
- Some error handling features
  - EXCEPTION_IN and EXCEPTION_OUT

One Example
- Assembly of Modules



2 × denorm
+ 1 × fp_add
+ 1 × rnd_norm
= 1 × IEEE single precision adder

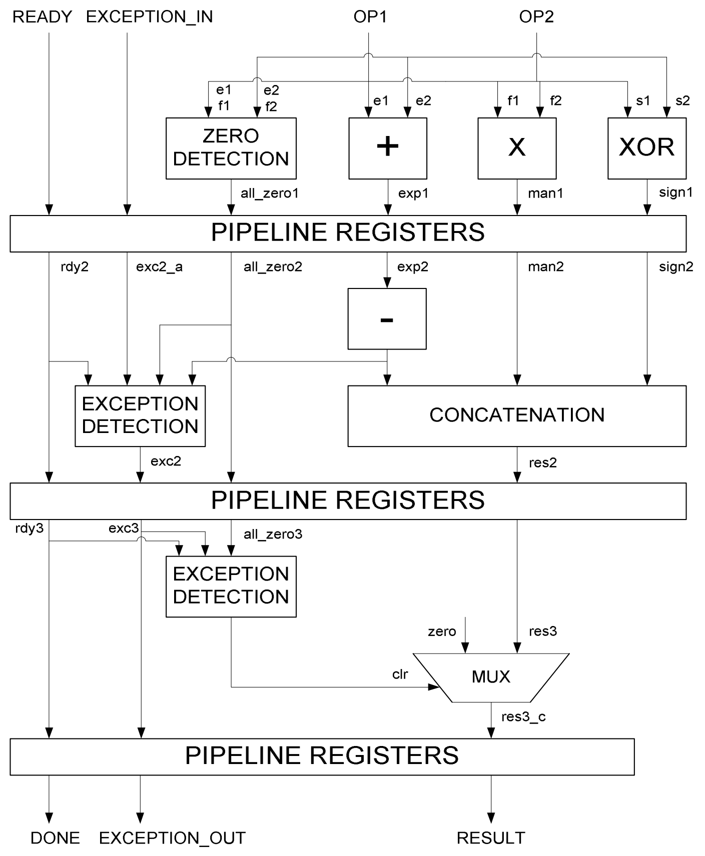
Another Example - Floating Point Multiplier



$$(-1)^{s1} * 1.f1 * 2^{e1\text{-}BIAS}$$

$$x \quad (-1)^{s2} * 1.f2 * 2^{e2\text{-}BIAS}$$

$$(-1)^{s1 \text{ xor } s2} * (1.f1*1.f2) * 2^{(e1+e2\text{-}BIAS)\text{-}BIAS}$$



Latency

Module	Latency (clock cycles)
denorm	0
rnd_norm	2
fp_add / fp_sub	4
fp_mul	3
fp_div	14
fp_sqrt	14
fix2float(unsigned/signed)	4/5
float2fix(unsigned/signed)	4/5

Clock rate of each module is similar

Outline

- Project overview
- Library hardware modules
- Floating point divider and square root
- K-means clustering application for multispectral satellite images using the library
- Conclusions and future work

Algorithms for Division and Square Root

- Division
  - P. Hung, H. Fahmy, O. Mencer, and M. J. Flynn, “Fast division algorithm with a small lookup table," *Asilomar Conference*,1999

- Square Root
  - M. D. Ercegovac, T. Lang, J.-M. Muller, and A. Tisserand, “Reciprocation, square root, inverse square root, and some elementary functions using small multipliers," *IEEE Transactions on Computers,* vol. 2, pp. 628-637, 2000

Why Choose These Algorithms?

- Both algorithms are simple and elegant
  - Based on Taylor series
  - Use small table-lookup method with small multipliers
- Very well suited to FPGA implementations
  - BlockRAM, distributed memory, embedded multiplier
  - Lead to a good tradeoff of area and latency
- Can be fully pipelined
  - Clock speed similar to all other components

Division Algorithm

Dividend X and divisor Y are 2m-bit fixed-point number $\in$ [1,2)

$$X = 1 + 2^{-1} x_1 + 2^{-2} x_2 + \ldots + 2^{-(2m-1)} x_{2m-1}$$

$$Y = 1 + 2^{-1} y_1 + 2^{-2} y_2 + \ldots + 2^{-(2m-1)} y_{2m-1}$$

,where $x_i, y_i \in \{0,1\}$

Y is decomposed into higher order bit part Y_h and lower order bit part Y_l , which are defined as

$$Y_h = 1 + 2^{-1} y_1 + 2^{-2} y_2 + \ldots + 2^{-m} y_m$$

$$Y_l = 2^{-(m+1)} y_{m+1} + \ldots + 2^{-(2m-1)} y_{2m-1}$$

,where $\boxed{Y_h > 2^m \bullet Y_l}$

Division Algorithm – Continue

Using Taylor series

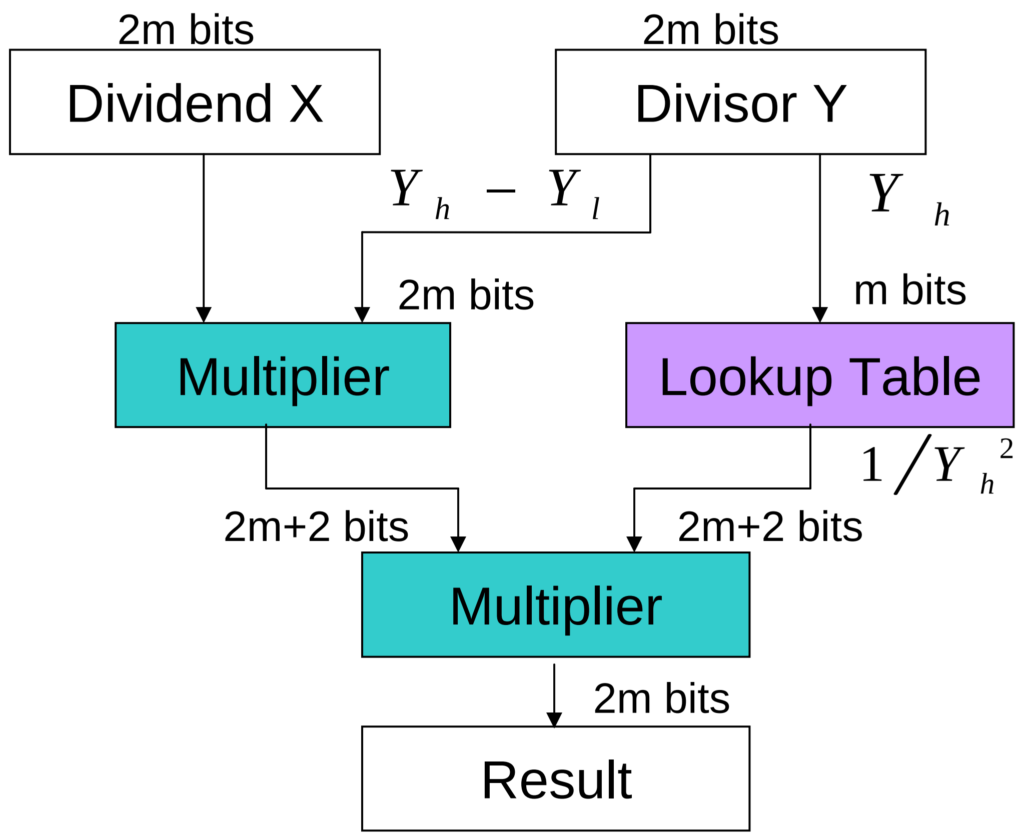
$$\frac{X}{Y} = \frac{X}{Y_h + Y_l} = \frac{X}{Y_h}\left(1 - \frac{Y_l}{Y_h} + \frac{Y_l^{\,2}}{Y_h^{\,2}} - \ldots\right)$$

$$\approx X \boxed{\times} (Y_h - Y_l) \boxed{\times} \boxed{\frac{1}{Y_h^{\,2}}}$$

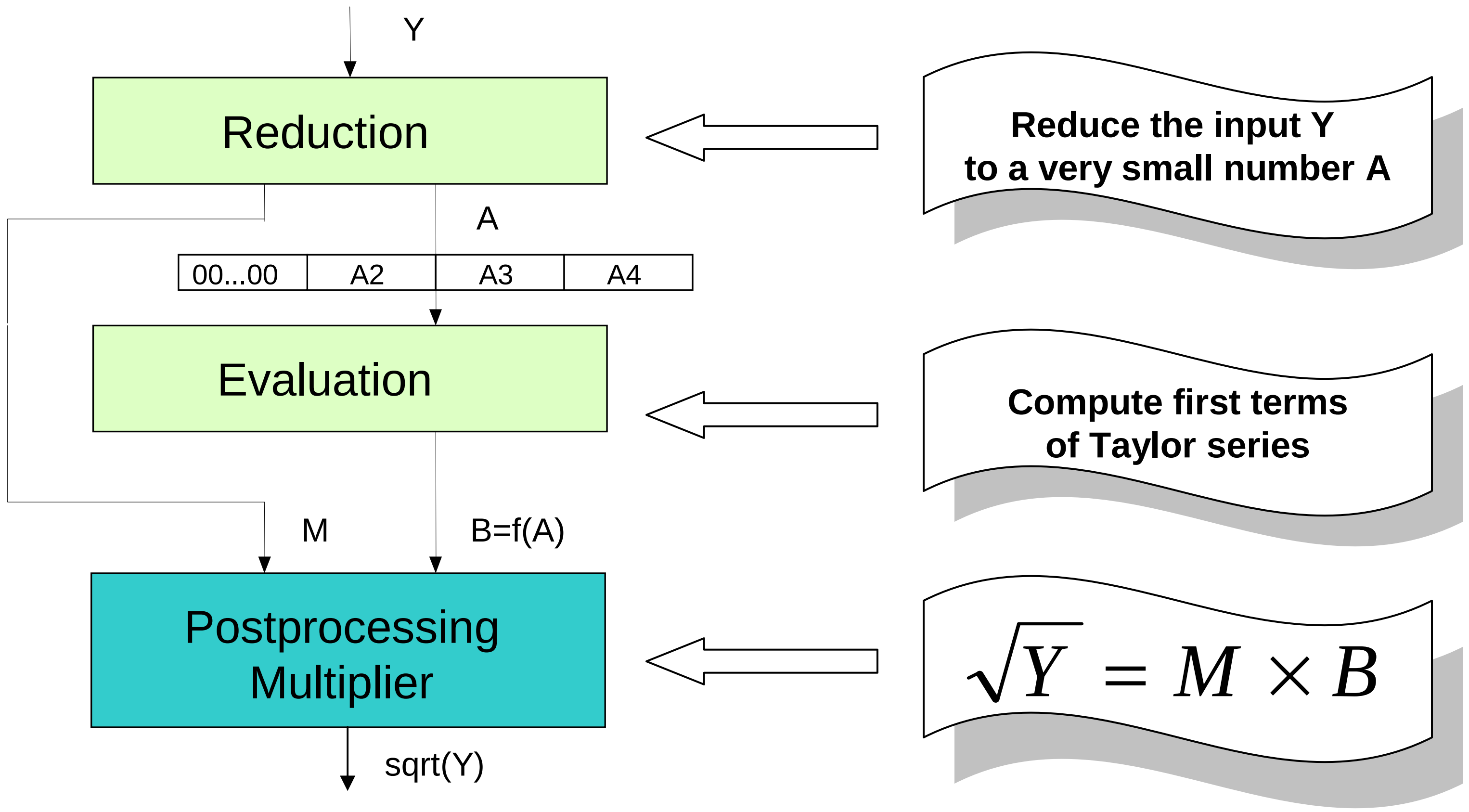
Error less than ½ ulp

Two multipliers and one Table-Lookup are required

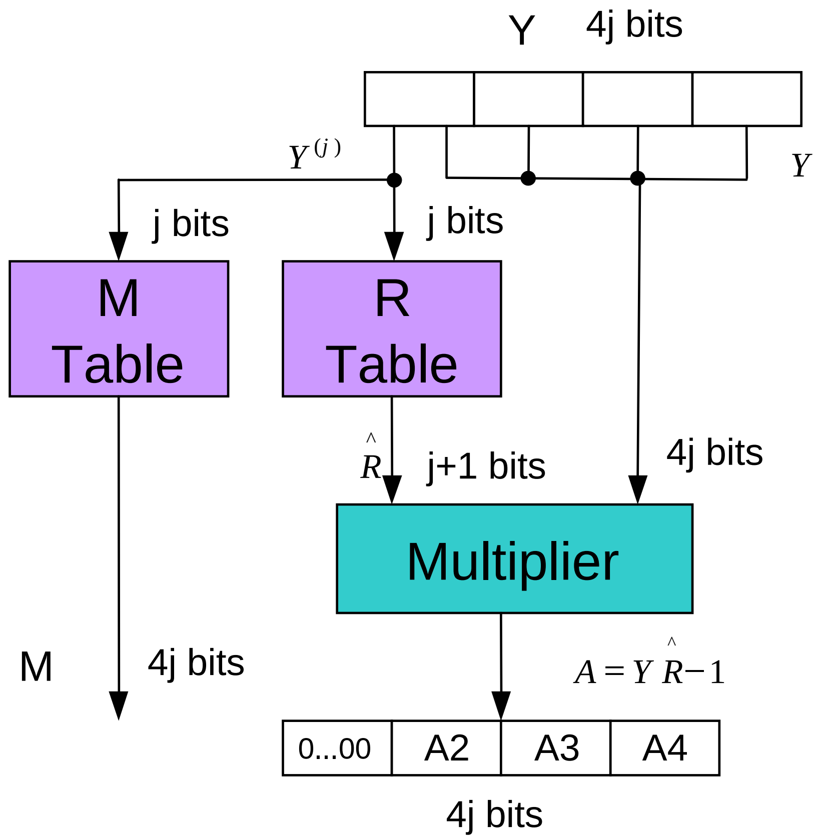
Division – Data Flow



Square Root – Data Flow


Y
Reduction
A
00...00
A2
A3
A4
Evaluation
M
B=f(A)
Postprocessing Multiplier
sqrt(Y)
Reduce the input Y to a very small number A
Compute first terms of Taylor series
$\sqrt{Y} = M \times B$

Square Root – Reduction

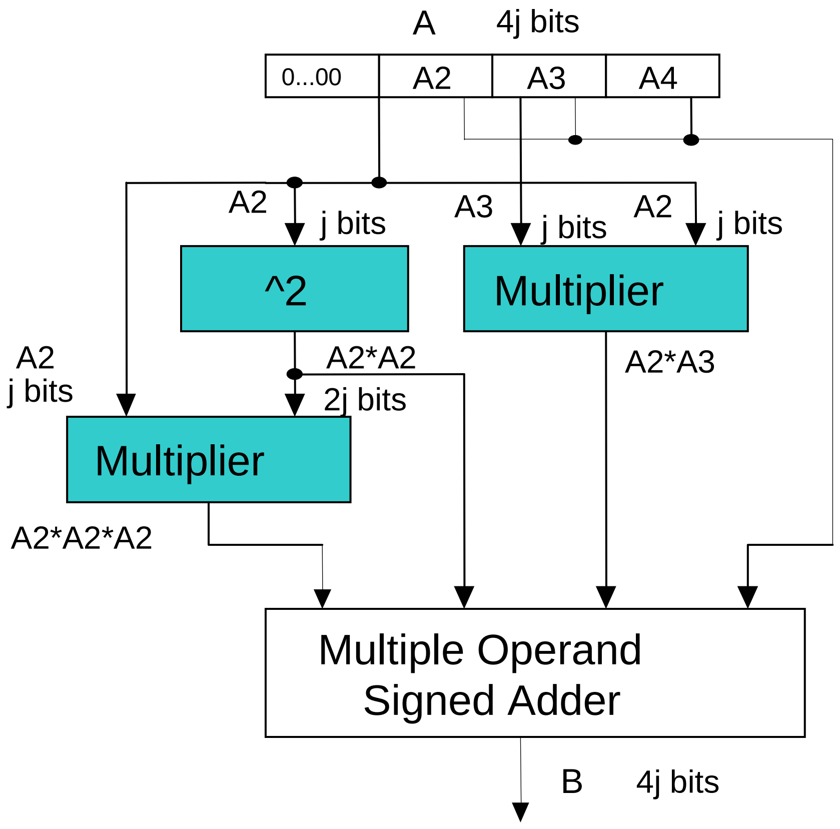


$$\hat{R} = 1 / Y^{(j)}$$

$$M = 1 / \sqrt{\hat{R}}$$

$$A = Y \times R - 1$$

Square Root - Evaluation

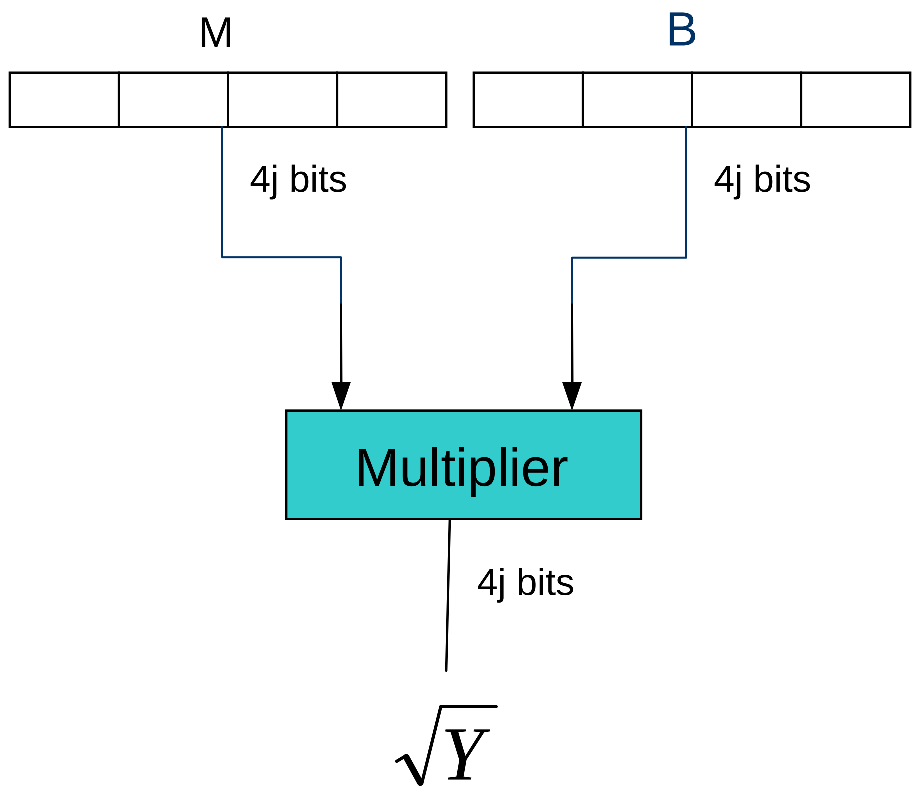


$$A = A_2 z^2 + A_3 z^3 + A_4 z^4 \ldots$$

$$B = \sqrt{1+A}$$

$$= 1 + \frac{2}{A} - \frac{1}{8} A_2^2 z^4 - \frac{1}{4} A_2 A_3 z^5 + \frac{1}{16} A_2^3 z^6$$

Square Root – Post Processing



$$\sqrt{Y} = M \times B$$

Results Mapping to Hardware

- Designs specified in VHDL
- Mapped to Xilinx Virtex II FPGA (XC2V3000)
  - System clock rates up to 300 MHz
  - Density up to 8M system gates
  - 14,336 slices
  - 96 18x18 Embedded Multipliers
  - 96 18Kb BlockRAM (1,728 Kb)
  - 448 Kb Distributed Memory
- Currently targeting Annapolis Wildcard-II

Results - FP Divider on a XC2V3000

Floating Point Format	**8(2,5)**	**16(4,11)**	**24(6,17)**	**32(8,23)**
# of slices	69 (1%)	110 (1%)	254 (1%)	335 (2%)
# of BlockRAM	1 (1%)	1 (1%)	1 (1%)	7 (7%)
# of 18x18 Embedded Multiplier	2 (2%)	2 (2%)	8 (8%)	8 (8%)
Clock period (ns)	8	10	9	9
Maximum frequency (MHz)	124	96	108	110
# of clock cycles to obtain final results	10	10	14	14
Latency (ns)=clock period x # of clock cycles	80	105	129	127
Throughput (million results/second)	124	96	108	110

The last column is the IEEE single precision floating point format

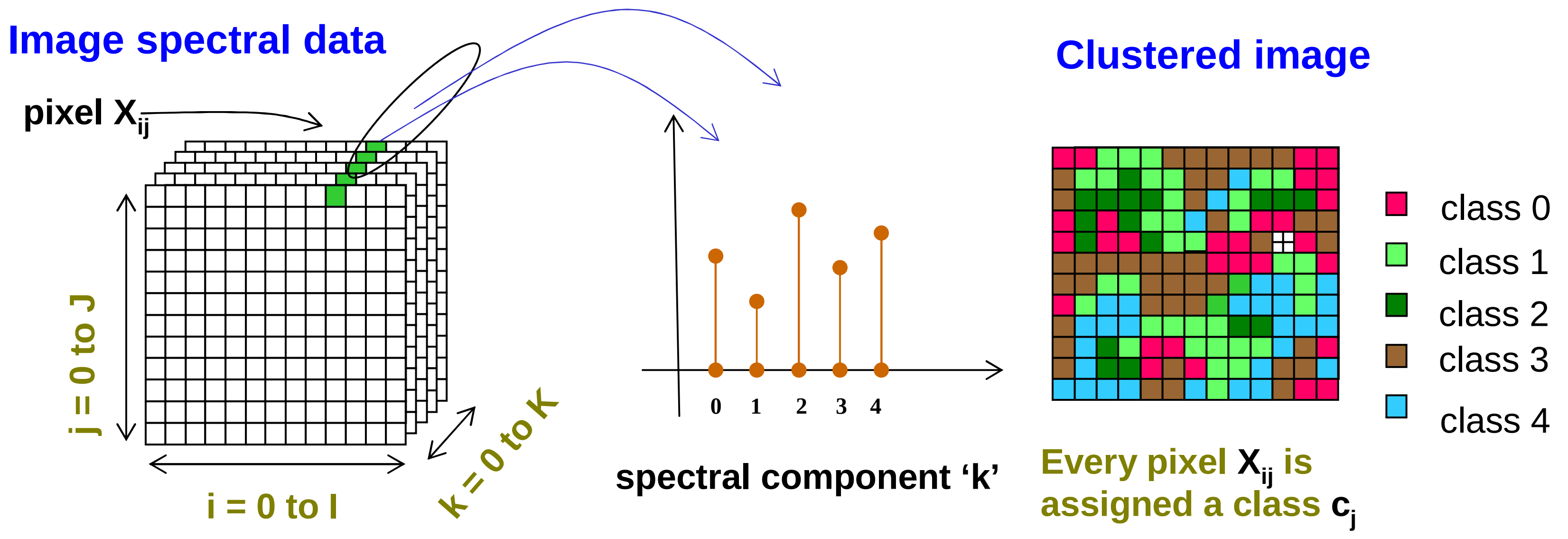
Results - FP Square Root on a XC2V3000

Floating Point Format	**8(2,5)**	**16(4,11)**	**24(6,17)**	**32(8,23)**
# of slices	113 (1%)	253 (1%)	338 (2%)	401 (2%)
# of BlockRAM	3 (3%)	3 (3%)	3 (3%)	3 (3%)
# of 18x18 Embedded Multiplier	4 (4%)	5 (5%)	9 (9%)	9 (9%)
Clock period (ns)	10	9	11	12
Maximum frequency (MHz)	103	112	94	86
# of clock cycles to obtain final results	9	12	13	13
Latency (ns)=clock period x # of clock cycles	88	107	138	152
Throughput (million results/second)	103	112	94	86

Outline

- Project overview
- Library hardware modules
- Floating point divider and square root
- K-means clustering application for multi-spectral satellite images using the library
- Conclusions and future work

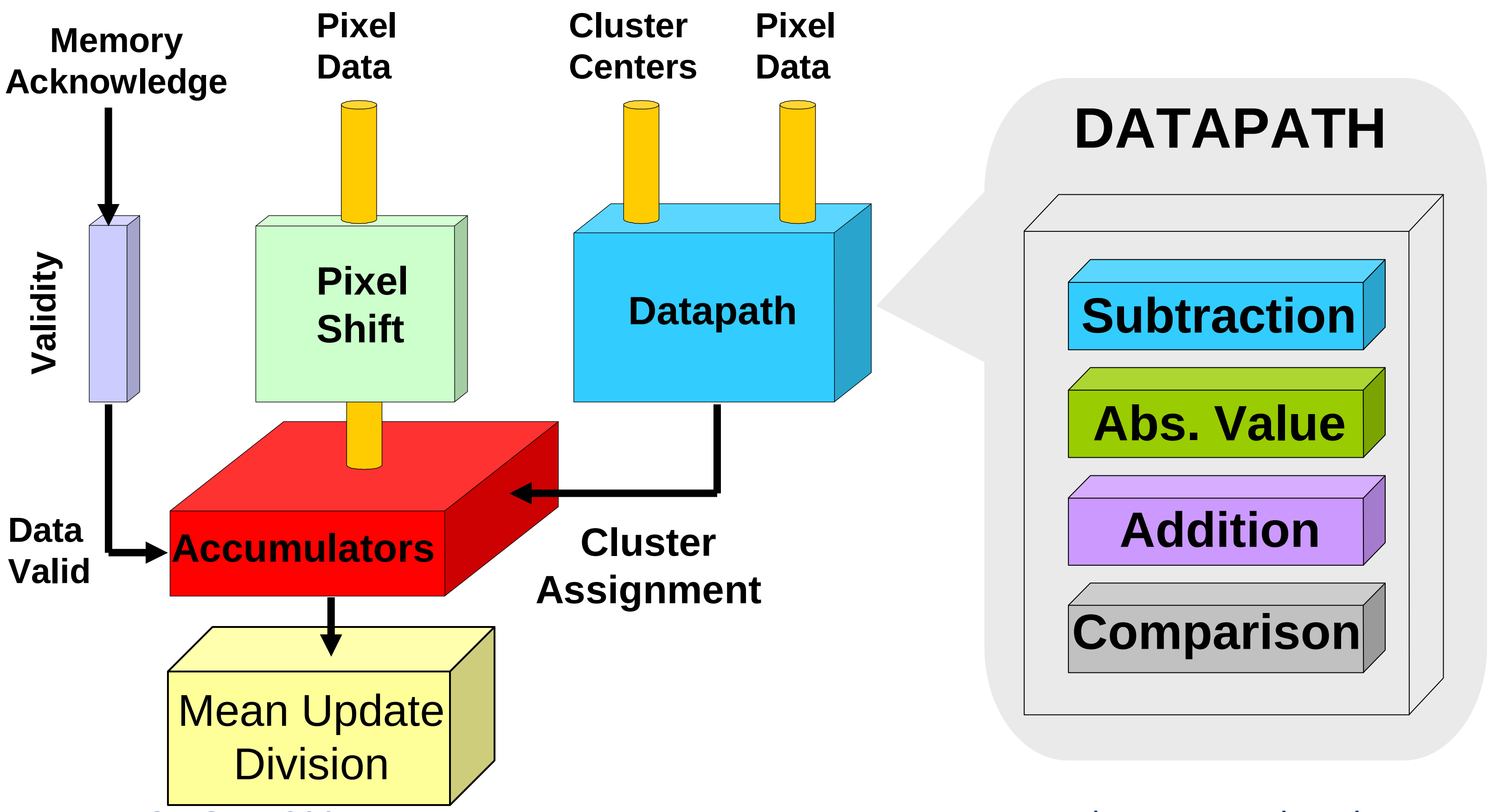
Application : K-means Clustering for Multispectral Satellite Images



K-means – Iterative Algorithm

- Each cluster has a center (mean value)
  - Initialized on host
  - Initialization done once for complete image processing
- Cluster assignment
  - Distance (Manhattan norm) of each pixel and cluster center
- Accumulation of pixel value of each cluster
- Mean update via dividing the accumulator value by number of pixels
- **Division step now executed on-chip with fp_divide to improve performance**

K-means Clustering – Functional Units



Outline

- Project overview
- Library hardware modules
- Floating point divider and square root
- K-means clustering application for multispectral satellite images using the library
- Conclusions and future work

Conclusion

- A Library of fully pipelined and parameterized hardware modules for floating point arithmetic
- Flexibility in forming custom floating point formats
- New module fp_div and fp_sqrt have small area and low latency, are easily pipelined
- K-means clustering algorithm applied to multispectral satellite images makes use of fp_div

Future Work

- More applications using
  - fp_div and fp_sqrt
- New library modules
  - ACC, MAC, INV_SQRT
- Use floating point lib to implement floating point coprocessor on FPGA with embedded processor

For Additional Information

Rapid Prototyping Laboratory

Northeastern University, Boston MA

http://www.ece.neu.edu/groups/rpl/

aconti , xjwang , mel @ece.neu.edu