Toward hardware support for Reproducible BLAS

http://bebop.cs.berkeley.edu/reproblas/

James Demmel, **Hong Diep Nguyen**

SCAN 2014 - Wurzburg, Germany

Sep 24, 2014

Reproducibility

Reproducibility: obtaining bit-wise identical results from different runs of the program on the same input data, regardless of different available resources.

Cause of nonreproducibility: *not* by roundoff error but by the *non-determinism* of accumulative roundoff error.

Due to the *non-associativity* of floating point addition, accumulative roundoff errors depend on the order of evaluation, and therefore depend on available computing resources.

Reproducible Summation

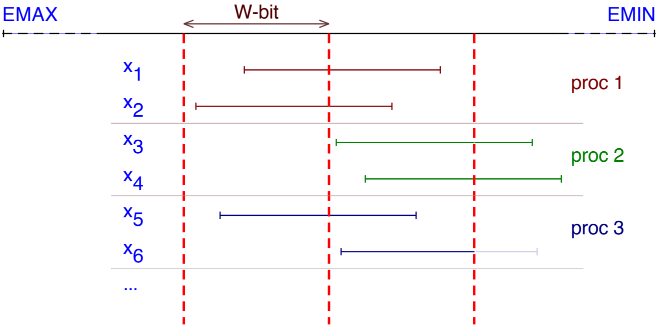
$$s = \sum_{1}^{N} v[i]$$

Error bound: $\left| s - \sum_1^N v[i] \right| < N \times \epsilon \times \sum_1^N |v[i]|$.
Running error depends on the order of evaluation.

Solutions:

- Increasing the accuracy (Kahan's algorithm, distillation algorithm, extra precision, ...) can increase the chance of reproducibility but does not guarantee reproducibility.
- Exact arithmetic or correctly-rounded algorithm can provide reproducibility: costly both in terms of memory and computation
- Our proposed solution: pre-rounding technique

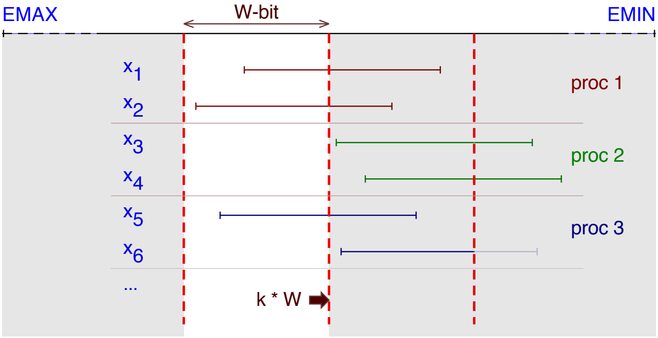
Reproducible Summation: Pre-rounding technique



1

[1] S.M. Rump, *Ultimately Fast Accurate Summation*, SIAM Journal on Scientific Computing (SISC), 2009

Reproducible Summation: Pre-rounding technique



1

[1] S.M. Rump, *Ultimately Fast Accurate Summation*, SIAM Journal on Scientific Computing (SISC), 2009

Indexed Floating-Point Format

Idea: representing the partial sum by:

- the index of the left-most bin:

$$\texttt{width}(\texttt{Index}) \geq \log_2\left(\frac{\texttt{EMAX} - \texttt{EMIN}}{\texttt{W}}\right)$$

- $\texttt{K}$ numbers of $\texttt{BW}$ bits to represent the $\texttt{K}$ left-most bin. Maximum number of addends that can be added without overflow:

$$N_{max} = 2^{\texttt{BW}-\texttt{W}-1-1}$$

- Absolute error bound:

$$|S - \sum_{1}^{N} v_i| \leq N \times \texttt{ulp}(\mathit{last\ bin}) = N \times 2^{-(\texttt{K}-1)\texttt{W}} \times \max_{1}^{N} |v_i|$$

ReproBLAS http://bebop.cs.berkeley.edu/reproblas

ReproBLAS is a library for (Parallel and Sequential) Reproducible Basic Linear Algebra Subroutines, currently only supports level-1 routines for 4 basic data type (single/double precision, real/complex numbers)

Configuration for double precision: $W = 40$, $K = 3$

- Can accumulate up to $2^{(P-1)-W-1} = 2^{11} = 2048$ numbers in mantissa part without any overflow.
- Can accumulate $2^{2*P-W-2} = 2^{64}$ numbers using carry part.
- The absolute error bound in the worst case is $N \times 2^{(K-1)*W} \times \max|v_i| = 2^{-80} \times N \times \max|v_i|$
- **Require only one reduction operation**.
- Run **8**× slower than performance-optimized library on a single processor, but only **1.2**× slower on massively parallel environment such as CRAY XC30 machine with 1024 processors.

Hardware support

Goals:

- Reduce the slowdown of reproducible operations on single processor to as close to $1\times$ as possible,
- Require minimal changes to current hardware,

Approaches:

- Dedicated Accumulator
- New instructions to support the implementation of reproducible addition:
  - Using existing 128-bit/256-bit register to represent indexed floating-point format,
  - Using existing load-store instructions,
  - Can be pipelined, multi-threaded.

Instructions

- Addition
  - a native floating-point to an indexed floating-point number
  - two indexed floating-point numbers
- Conversion
  - From native floating-point number to indexed numbers: implicitly through the addition
  - From indexed format to native format: not frequently used, can be implemented in software
- Carry-bit propagation: propagate the overflow bit to a higher order register to increase the maximum number of addends.

Data Format Layout

Requirements:

- $\texttt{width}(\texttt{Index}) + \texttt{K} \times \texttt{BW} \leq \texttt{register width}$
- $\texttt{BW} > \texttt{W}$
- $\texttt{width}(\texttt{Index}) \geq \log_2\left(\frac{\texttt{EMAX}-\texttt{EMIN}}{\texttt{W}}\right)$
- Reasonable error bound:

$$|S - \sum_{1}^{N} v_i| \leq N \times 2^{-(\texttt{K}-1)\texttt{W}}$$

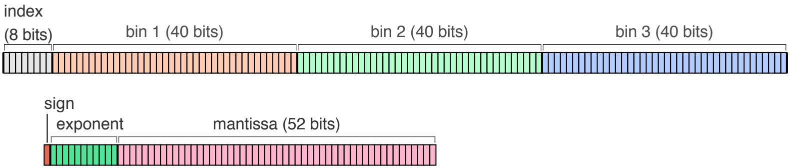
For double precision floating-point number, using 128-bit register:

- $\texttt{width}(\texttt{Index}) + \texttt{K} \times \texttt{BW} \leq 128$
- $\texttt{width}(\texttt{Index}) \geq 11 - \lceil \log_2(W) \rceil$

Configuration: $K = 3$, $W = 32$, $\texttt{BW} = 40$, $\texttt{width}(\texttt{Index}) = 8$

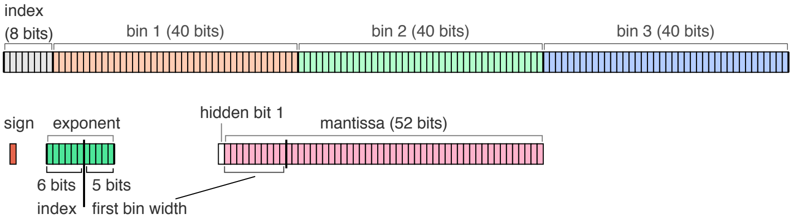
128-bit Indexed Floating-Point Format

Configuration: $K = 3$, $W = 2^5$, $\texttt{BW} = 40$, $\texttt{width(I)} = 8$



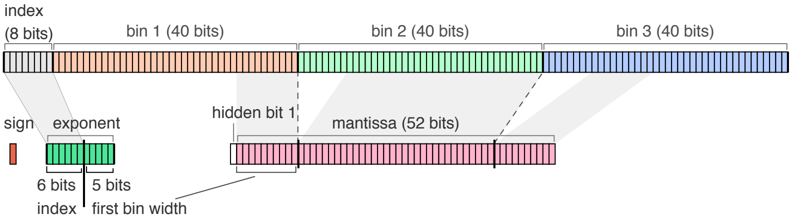
128-bit Indexed Floating-Point Format

Configuration: $K = 3$, $W = 2^5$, $\mathtt{BW} = 40$, $\mathtt{width(I)} = 8$



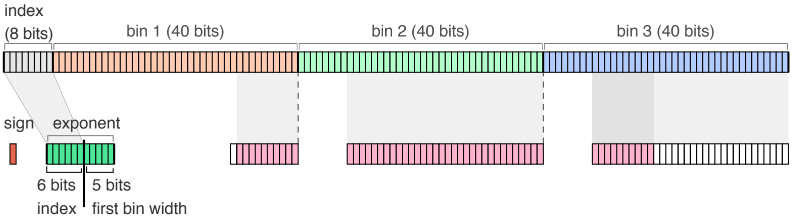
128-bit Indexed Floating-Point Format

Configuration: $K = 3$, $W = 2^5$, $\texttt{BW} = 40$, $\texttt{width(I)} = 8$



128-bit Indexed Floating-Point Format

Configuration: $K = 3$, $W = 2^5$, $\texttt{BW} = 40$, $\texttt{width(I)} = 8$



Properties

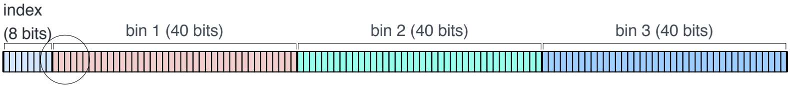
Absolute error bound in the worst case:

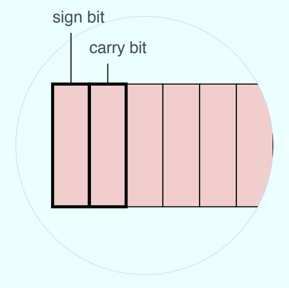
$$|s - \sum_{1}^{N} v[i]| \leq N \times 2^{(K-1)*W} \times \max|v[i]| = 2^{-64} \times N \times \max|v[i]|$$

Number of additions that can be performed without overflow:

$$N_{max} = 2^{40-32-1-1} = 64$$

Carry Propagation





(sign bit, carry bit) = (0,0) : c = 0
(sign bit, carry bit) = (0,1) : c = 1
(sign bit, carry bit) = (1,0) : c = -2
(sign bit, carry bit) = (1,1) : c = -1

Objective: ensure that there will be no overflow over the next $2^{\mathtt{BW}-\mathtt{W}-1-1}$ additions. Using the same format for the carry register:

$$N_{max} = 2^{\mathtt{BW}-\mathtt{W}-1-1} * 2^{\mathtt{BW}-2} = 2^{45}$$

Experimental results

Simulation in software:

- Implemented in Chisel, a scala-based programming language for hardware construction.
- Operations:
  - `RAdd`: add 1 double precision FP to an 128-bit Indexed Floating-Point,
  - `RAddR`: add 2 128-bit Indexed Floating-Point,
  - `RRenorm`, `RCarry`: perform bit propagation to avoid overflow.

  each operation can be executed in 1 clock cycle
- No exception-handling
- $\approx$ 230 LOC for the hardware construction
- $\approx$ 400 LOC for testing and validation

Reproducible Summation: Algorithm

Sequential summation of N double precision floating-point numbers

```
int i, NB = 64;
Idouble s, c;
for (iN = 0; iN < n; iN += NB) {
   for (i = iN; i < min(n, i+NB); i++) {
      s = RAdd(s, v[i]);
   }
   c = RCarry(c, s);
   s = RRernorm(s);
}
```

Cost: $N + \mathcal{O}(\frac{N}{NB})$ FLOPs

Reproducible Sum: Accuracy

$$v[i] = \sin(2.0 * Pi * i/N), \quad N = 10^5$$

Algorithm	$1 \rightarrow N$	$N \rightarrow 1$
quadruple	**9.92341383715**7274E-15	**9.92341383715**682E-15
Reproducible	**9.923**377224108076E-15	**9.923**377224108076E-15
Normal Sum	5.513115788968589E-13	3.0460904930145957E-12

Conslusion

New instructions:

- Operates on existing 128-bit register file,
- Executes in 1 single cycle,
- Requires no change to the scheduling system,
- Helps to reduce the cost of reproducible summation to $\approx N$ FLOPs, and is almost as accurate as the normal summation algorithm.

TODO

- Implementation on real hardware to collect real data on required area as well as the energy consumption of proposed instructions.
- Fused Multiply-Add support
- Implementation for hardware without support of 128-bit register.
- Implementation of BLAS level 2, 3 routines.
- Implementation of software library that provides exactly the same results as those computed using the newly proposed instructions.