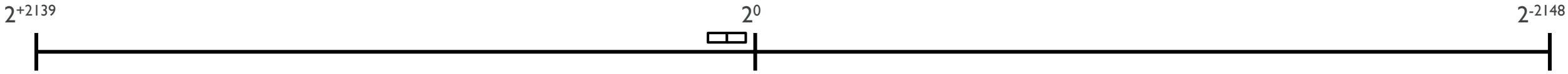
High-Precision Anchored Accumulators for Reproducible Floating-Point Summation

ARM

David Lutz and Neil Burgess

Kulisch accumulators

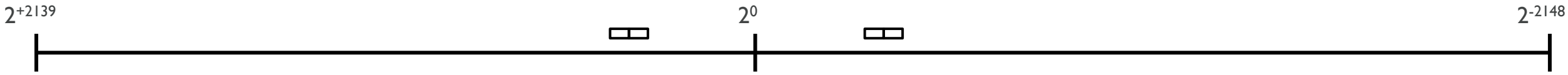
- treat FP numbers and products as very wide fixed-point numbers
- use a 4288-bit accumulator!
  - 4092 possible locations for first significand bit
  - 105 fraction bits
  - extra bits so as to avoid overflow
  - 67 64-bit words
- these additions are **associative**, and the 4288-bit result is **exact**



ARM

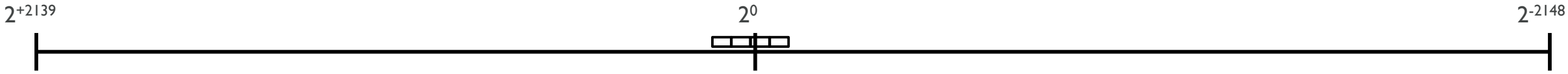
Reducing exponent range

- most problems do not require the full exponent range
  - galaxies or subatomics?
- many programmers use FP for convenience
- small values may well be unimportant
- programmers can know & benefit from knowing these ranges



What is a typical range?

- 100 bits suffices for many HPC applications" (D. Bailey, 2013 ARITH keynote)
- "most problems fit in the range 10^{-25} to 10^{30}, a span of about 183 bits" (LANL)
- "128-bit integers are probably sufficient for most uses." (LANL again, SC15)
- "... in most cases we're around the 10^{-15} tolerance [2^{-50}] because of machine epilson, compiler rounding/optimization etc with results in a tighter range with lower exponents. (Sandia)



Where are we going to get 100 or 200 bit integers?

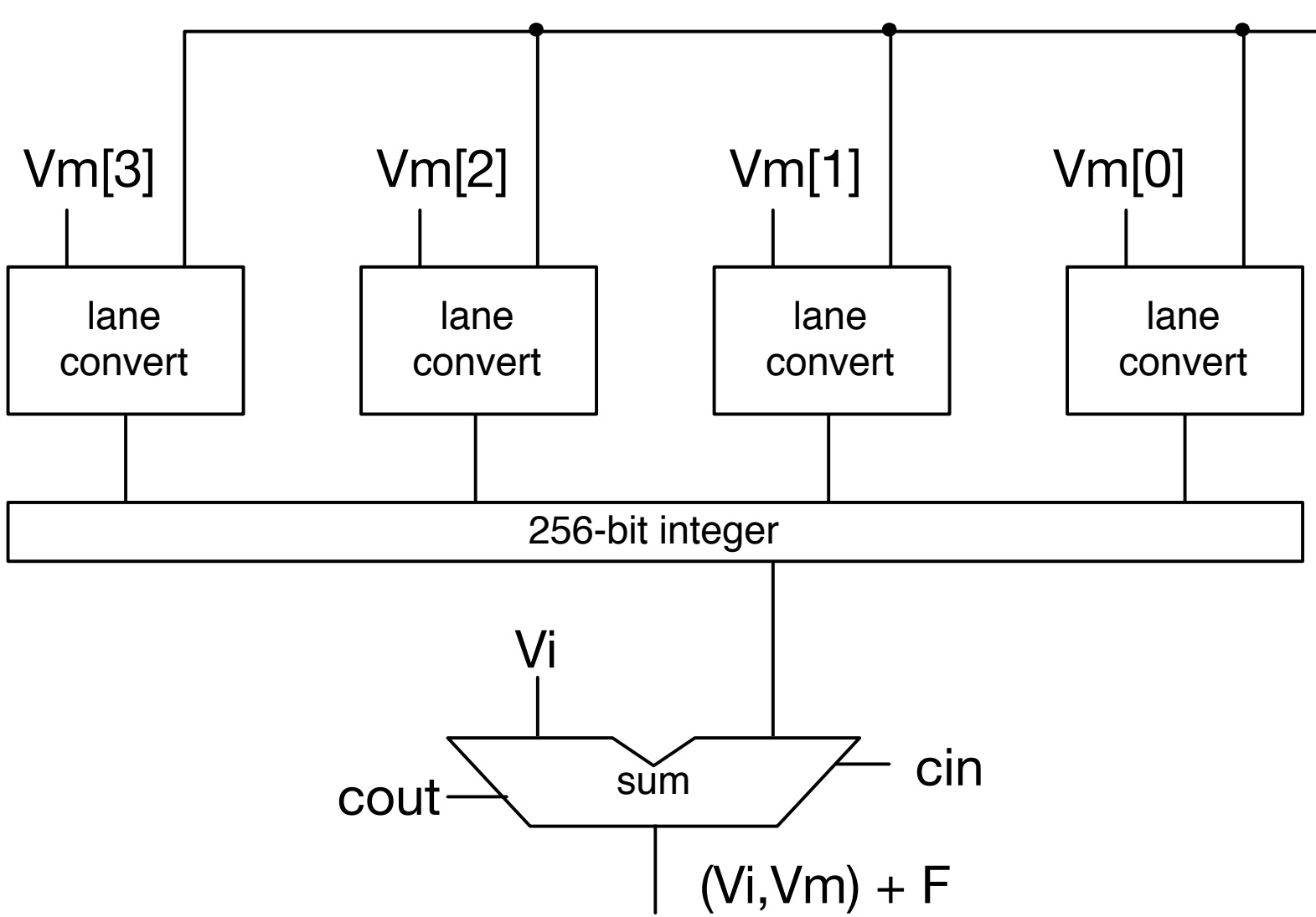
- SIMD units are close approximations
  - Central concept: treat vector of 64-bit values as one long integer
  - ARM NEON is 128 bits
  - ARM SVE (just announced) 128 to 2048 bits

High-Precision Anchored (HPA) Numbers

- An HPA number comprises:
  - a long 2's-complement integer, containing 100-200 (or more) bits
  - an anchor that says how to interpret those bits

- a programmer picks the range for the application area or problem
  - anchor is analogous to a floating-point exponent, but is fixed for a given problem
  - anchor represents the least significant exponent value we are interested in
  - the length of the long integer then gives us a range over which we can accumulate exactly
- HPA accumulation is associative, reproducible, and parallelizable

Adding, Subtracting, Converting FP to HPF



ADD_HPA_FP (Vi,Vm, F)

SUB_HPA_FP (Vi,Vm, F)

CVT_HPA_FP (Vi,Vm, F)

FP Accumulation

- FP2HPA convert & add is a 2-cycle latency, fully pipelined operation on CPUs
- add n (or 2n) items in n+1 cycles, vs. adding n items in 3n cycles for A72 Neon
- establishing exponent range is the only additional task for a programmer
- these adds are associative, so no dependencies ∴ fully parallelizable

	1	2	3	4	5
ADD_HPA_FP (Vi,Vm,F1)	Convert	Add			
ADD_HPA_FP (Vi,Vm,F2)		C	A		
ADD_HPA_FP (Vi,Vm,F3)			C	A	
ADD_HPA_FP (Vi,Vm,F4)				C	A

Sum of FP products

- given HPA number (Vi,Vm) and FP numbers F1 and F2
- MUL_HPA_FP (Vi,Vm, F1, F2): store F1*F2 as an HPA number
  - compute F1 * F2 without rounding
  - each lane gets a copy of the unrounded product (or computes the product) and Vm
    - A product will span more lanes than an FP64 number
  - unrounded product converted to HPA using same technique as for FP numbers
  - fully pipelined
- MAC_HPA_FP (Vi,Vm, F1, F2): add F1*F2 to an HPA number

What If the Anchor Range is Wrong?

- on the low end, some numbers may convert as zeros or lose accuracy
  - this could be a deliberate choice to avoid insignificant data
  - addition is still associative, parallelizable, and reproducible in this case
- on the high end, conversion will signal overflow
  - this is a problem that needs to be fixed
  - set an ovf flag? trap?
  - ... OR scan input set for maximum value

Simple Programming Model

- need to pick an expected exponent range (or scan data set for max. value)
- set up exponent base value(s) covering that range
  - software library could make it even simpler
- convenience of FP without the problems of FP
  - suitable for NEON or SVE
- no need to restrict numbers to FP accuracy
  - e.g., ≥128-bit accurate π could be useful in range reduction

Paradigm Shift?

- "the fast drives out the slow even if the fast is wrong." - W. Kahan
  - but what if the fast is right?
- why deal with the irreproducibility & incorrectness of FP accumulation?
- FP accumulation that is reproducible, parallelizable, faster, and correct

- Is this approach useful?

Imperial College, LONDON

24-26 July 2017

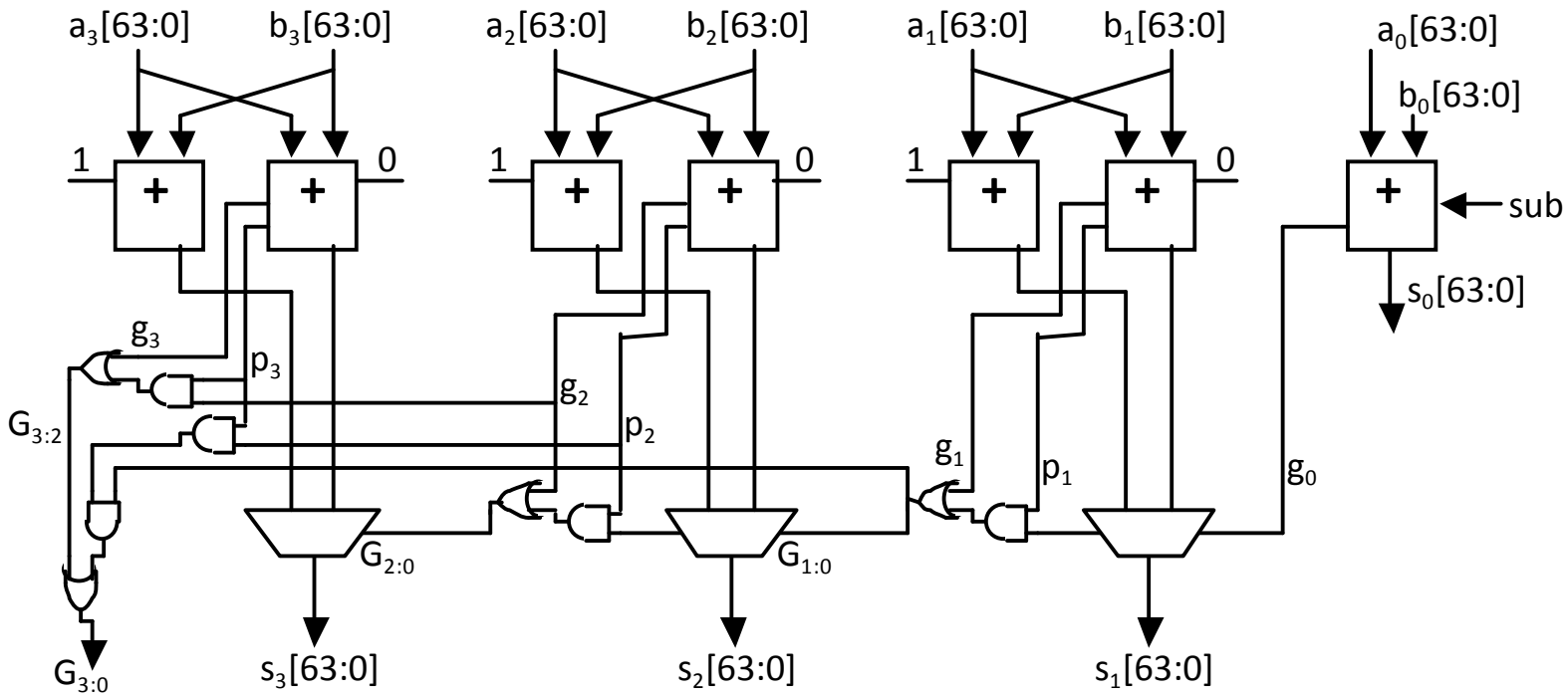
Submission deadline: 31 December 2016

www.arith24.arithsymposium.org

Extra 1: why are we using double precision?

- Single precision: around 10^{-38} to 10^{38}
- Do measurements have more than 24 significant bits of accuracy?
- My guess: we use DP because of associativity problems
- HPA would allow us to use SP:
  - double memory bandwidth
  - double computation bandwidth
  - half the power per flop
  - exact, reproducible sums and sums of products

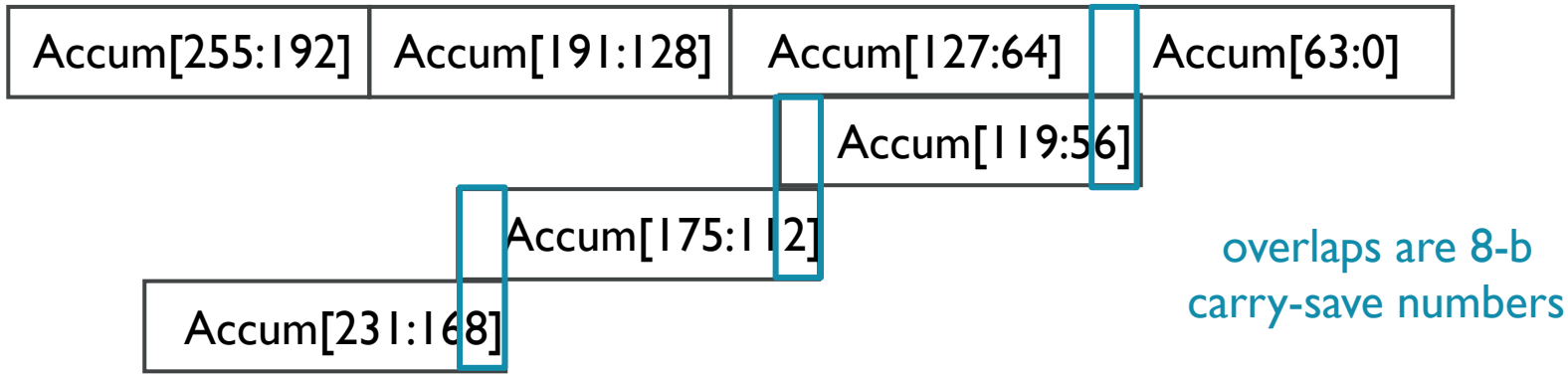
Extra 2: multi-lane addition/subtraction



- Possible, but not ideal for SIMD paradigm
- Requires cross-lane carries

Redundant Long Integer Arithmetic

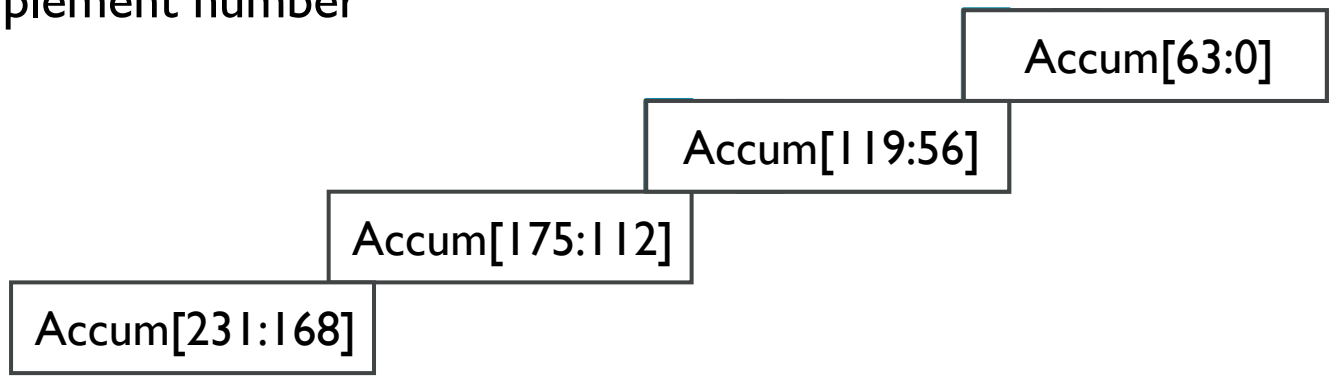
- Allow vector elements to "overlap"
- For example, allowing 8 bits' overlap between lanes:



- Provide headroom in each lane to accommodate carries
- Treat each lane as a 2's-complement number

Redundant Long Integer Arithmetic

- Can complete $2^8 - 1 = 255$ additions/subtractions without carries needing to transfer between lanes
- Periodically, need to "reset" carries
- Set to all 0's by sequential addition of overlap bits from lower lane to next higher lane
  - Full-width 2's-complement number



- Alternative parallel technique restricts overlap values to {+1, 0, -1}