Systolic Super Summation with Reduced Hardware

Willard L. Miranker

Mathematical Sciences Department

IBM T.J. Watson Research Center

Route 134 & Kitichwan Road

Yorktown Heights, NY 10598

Abstract

A principal limitation in accuracy for scientific computation performed with floating-point arithmetic may be traced to the computation of repeated sums, such as those which arise in inner products. As previously reported, a systolic super summer is a cellular piece of hardware for the summation of floating-point numbers. The apparatus receives floating-point summands, converting them into a fixed-point form within a sieve-like cellular array. The emerging fixed-point numbers then are summed in a pipelined array of long accumulators. An improved design is presented for systolic super summation's sieve. Although the new sieve is structurally simpler and uses less hardware, the throughput per unit area is the same as the previously designed sieve. The architectural regularity of the new sieve makes it ideal for implementation in VLSI circuit technology.

1 Introduction

1.1 Super summation

An operation called the *floating-point inner product* (FLIP), has been advocated [7, 8] in order to eliminate the need for scaling while still enjoying error-free, associative accumulation. Such an operation can be simulated by iterative algorithms [7, Ch. 6], [4, 5], and parallel algorithms [9]. High performance hardware units also have been devised [1, 6]. The unit described in [1] has been more or less implemented, by means of microcoded assists, in the IBM 4361. This IBM unit uses a *super* (i.e., very long) *accumulator*, the output of which is called a *super summation*.

In [3], we provide a design called systolic super summation (SSS) for the cellular summation of floating-point numbers. Some additional design details are furnished in [2]. We propose here a modification of that design which uses substantially less hardware (approximately an order of magnitude less for the design parameters that we have in mind). The sieve is the dominant component of the design with respect to area. The throughput per unit area for the proposed sieve is asymptotically equal to that of the sieve in [3]. The proposed design thus provides an area vs. throughput tradeoff. We argue in the conclusion that the proposed design is a more realistic choice for integration into a scientific computing system.

1.2 The design of Cappello and Miranker

We provide an overview of the design presented in [3]. Let $R(b, M, e1, e2)$ denote the collection of floating-point numbers with base b, mantissa length M, and minimum [maximum] exponent $e1$ [$e2$]. Suppose that a block of floating-point numbers in $R(2, M, 2e1, 2e2)$ representing summands arrive at this apparatus in bit-parallel order. The zero floating-point

number is appended to the end of a block in order to flush and initialize the apparatus. The output mantissa length is a device parameter called $\overline{M}$ where $\overline{M} < M$. Thus, each completed sum is a number in $R(2, \overline{M}, e1, e2)$,

A schematic of the systolic super summation device is given in Figure 1. Upon entry, the summands flow through a bit skewing device, emerging from it in serial-parallel order. This reordering of the bits is performed to accommodate bit collisions which occur in the sieve, the next part of the apparatus. The skewed summands enter the sieve. Each summand is accompanied by a bit, the end-of-sum signal (whose value is 'true' or 'false'). The sieve is a systolic packet switching network with signal combining. The mantissa of an entering summand is switched from its entry at the top of the sieve to an appropriate position at the bottom. The switching path is determined by the exponent of the summand. The appropriately switched summand mantissa emerging from the sieve enters a pipeline, called the A-pipeline, of accumulators, themselves denoted $\alpha_0, \alpha_1, \ldots, \alpha_{P+1}$ successively. These accumulators are able to accommodate each mantissa as if it were a fixed-point number, and the sieve is a shifting device for determining the correct fixed-point position for addition of the summands in α_0. Accumulator α_0, a super(-long) accumulator, is $M + 2^C$ bits long, where design parameter $C = \lfloor 2 + \log|e1| + \log e2 \rfloor$. In fact several, say F, overflow bit positions are appended to the left (the high-order end) of α_0 as well as all other accumulators in the A-pipeline.

The sum is accumulated in α_0. The end-of-sum signal bumps the contents of α_0 to α_1. Thus α_0 is initialized and ready for the next sum which may emerge from the sieve. In α_1 the carry resolution process in the summing continues, if necessary. This is combined with the encoding of the sum into a floating-point number in $R(2, \overline{M}, e1, e2)$. That is, as new sums arrive to bump predecessor sums down the A-pipeline, the latter are completing their carry resolution, shifting out leading zeros (normalizing), re-encoding the exponent,

and finally rounding; all of this more or less, concurrently.

Since sums may be completed in an order quite different from their order of entry into the super summer, the A-pipeline is furnished with a sum completion detection capability. As soon as a completed sum is detected, it is made to descend through the A-pipeline to be ejected from the apparatus. A completed sum may pass through other sums which are still in progress. This early sum departure feature, asynchronizes the process, and it is for this reason that sum tagging is introduced. An identity pointer (tag) enters the device with each summand block. The tag flows through the device in step with the summation process. The tag accompanies the sum as it departs the apparatus.

Entry of a sum into the sieve occupies only M of the $M + 2^C$ bit width of the sieve. The design thus is modified to increase entry capacity. The sieve is augmented into a so-called cylinder sieve (see Figure 2).

[3] Every disjoint band of width M becomes a port of entry into the sieve thereby increasing its bandwidth. There are now many streams of summand bits which descend in the sieve. These streams cross and collide. Collisions are resolved by combining bits (addition of bits) within the sieve itself. A key part of this design is the regulation of this bit traffic, and a proof that it works.

2 A design with reduced hardware

We present a design that uses less hardware than that presented in [3]. It is essentially the same as the design presented in [3], but replaces the large sieve with a much smaller one. The overall system is not cylindrical, because the sieve is not cylindrical. (see Figure 3). Although the new sieve is much smaller and simpler than the original sieve, the interface between the sieve and the super accumulator is unchanged.

Operands enter the sieve in a bit-skewed manner: An summand's characteristic bits

enter before its mantissa bits; high-order characteristic bits enter before low-order bits; high-order mantissa bits enter before low-order bits. A summand skewer (see Figure 3) can be used to skew the summands. Let us focus on the sieve, the new part of the overall design. Figure 4 illustrates its size and simple structure.

2.1 Characteristic decoding

The summand's characteristic, represented with C bits, ranges in value from 0 (i.e., C '0' bits) to $2^C - 1$ (i.e., C '1' bits). For each of these 2^C characteristic values, the sieve has a C-bit hard-wired characteristic *key*. Each summand starts out moving westerly through the sieve. When its characteristic matches one of the 2^C characteristic keys located along the top of the sieve, its mantissa starts moving southwesterly. Key matching is illustrated in Figure 5. Since the summand's characteristic is bit-skewed (from high-order bits to low-order bits), its C bits encounter the vertically aligned key matching cells on C successive cycles. The boolean variables for an individual key match cell are as follows:

- e is an exponent bit (i.e., a characteristic bit of a summand)
- k is a key bit (i.e., characteristic key bit)
- f is the flag bit (i.e., the match flag, a value of 1 indicates a match)

The equations governing cell input/output, as illustrated in Figure 5, are as follows[1]:

$$e \leftarrow e$$

$$f \leftarrow f \cdot (e \equiv k)$$

Initially, $f \leftarrow 1$. When and only when the summand characteristic matches a key, the flag (f) that is entering the mantissa portion of the sieve has value 1.

[1]The binary operation $\equiv$ means p is equivalent to q: $(p \cdot q) \vee (\overline{p} \cdot \overline{q})$

2.2 Mantissa movement

When the mantissa receives a key flag that is high (i.e., $f = 1$), its bits change the direction of their motion, in succession, from westerly to southwesterly. Since the summand's mantissa is bit-skewed (from high-order bits to low-order bits), its M bits encounter the vertically descending key flag on M successive cycles.

A mantissa bit that starts moving southwesterly can 'collide' with a bit of another mantissa that is moving southwesterly. In this case, both bits are destined for the same position of the super accumulator. We thus may add them, possibly producing a carry bit[2]. This logic is illustrated in Figure 6. The boolean variables are as follows:

- f is the flag bit (i.e., the match flag that emerges from the characteristic decode region)
- m is a mantissa bit (i.e., a mantissa bit of a summand)
- s is a sum bit (see Figure 6)
- c is a carry bit (see Figure 6)

The equations governing cell input/output, as illustrated in Figure 6, are as follows:

$$
\begin{aligned}
f &\leftarrow f \\
m &\leftarrow \overline{f} \cdot m \\
s &\leftarrow s \oplus c \oplus (f \cdot m) \\
c &\leftarrow (s \cdot c) + (s \cdot f \cdot m) + (c \cdot f \cdot m)
\end{aligned}
$$

The area, A, of the sieve is $O((C + M)(2^C + M))$. The minimum time for all the summands to enter the sieve (assuming there are at least 2 summands) is $C + M + 1$. So the

[2]The phenomenon of collison/co-habitation is illustrated and described in detail in [3], hence omitted from this note.

number of super accumulators, P, needed in a completely pipelined design (see [3]) is given by $\lceil (2^C + M)/(C + M + 1) \rceil$. Envisioned values for C and M are 9 and 48, respectively. For these values, $P = \lceil (2^9 + 48)/(9 + 48 + 1) \rceil = 10$.

2.3 Sum unloading

The sum unloading procedure is designed so that the sieve/accumulator hardware interface is the same as that reported in [3]. Unloading proceeds as follows.

1. One cycle after the least significant bit of the last summand enters the sieve, the sum *unload* signal (σ) is input with value '1'; on all other cycles, it is input with value '0'.

2. This unload signal moves systolically from right to left through the *unload* register (whose length matches that of a super accumulator). The unload signal also is directed out the bottom of the unload register cell. As in the original design, the unload signal enters the super accumulator cells from above.

3. An unload signal ($\sigma = 1$) causes the super accumulator cell to unload its sum and carry bits, and to initialize these variables to '0'.

4. Each super accumulator thus is unloaded systolically from least significant bit to most significant bit. Carry resolution consequently is completed as the super accumulator is unloaded.

3 Conclusion

In [3], we presented a design for the cellular summation of floating-point numbers. We proposed here a modification to that design's sieve which uses substantially less hardware. The sieve also is structurally less complicated, and accepts only one summand/cycle. Such

a device is much simpler to integrate into a computing system than the one presented in [3], whose input capacity (for envisioned parameter values) is 11 summands/cycle.

The two design's characteristics are compared in Table 1.

	Asymptotic value		**For $C = 9$ & $M = 48$**	
Measure	**Design [3]**	**Proposed design**	**Design [3]**	**Proposed design**
Area in bit-level sieve cells	$(2^C + M)^2(\frac{C+M}{M})$	$(C + M)(2^C + M)$	372,400	31,920
Throughput in summands/cycle	$(2^C + M)/M$	1	11	1
Area/Throughput	$(C + M)(2^C + M)$	$(C + M)(2^C + M)$	33,855	31,920

Table 1: *The design in [3] vs. the proposed design.*

We believe that the proposed design's

- structural simplicity,
- order of magnitude smaller size, and
- ease of system integration

are compelling advantages.

References

[1] G. Bohlender and U. W. Kulisch. Features of a hardware implementation of an optimal arithmetic. In U. W. Kulisch and Willard L. Miranker, editors, *A New Approach to Scientific Computation*. Academic Press, 1983.

[2] Peter R. Cappello and Willard L. Miranker. Systolic super summation. Technical Report RC 11826, IBM, April 1987.

[3] Peter R. Cappello and Willard L. Miranker. Systolic super summation. *IEEE Trans. Comput.*, 37(6):657–677, June 1988.

[4] IBM. *High Accuracy Arithmetic, Subroutine Library, General Information Manual*. Program Number 5664-185.

[5] IBM. *High Accuracy Arithmetic, Subroutine Library, Program Description and User's Guide*. Publication Number GC 33-6163.

[6] R. Kirchner and U. W. Kulisch. Arithmetic for vector processors. In M. J. Irwin and R. Stefanelli, editors, *Proc. 8th Symp. on Computer Arithmetic*, pages 256–269. IEEE Computer Society Press, Como, ITALY, May 1987.

[7] U. W. Kulisch and Willard L. Miranker. *Computer Arithmetic in Theory and Practice*. Academic Press, 1981.

[8] U. W. Kulisch and Willard L. Miranker. The arithmetic of the digital computer. *SIAM Review*, 28(1):1–40, Mar. 1986.

[9] H. Leuprecht and W. Oberaigner. Parallel algorithms for the rounding exact summation of floating point numbers. *Computing*, 28, 1982.

Figure 1: *Overall schematic of systolic super summation device .*

Figure 2: [a] *Summands enter the cylinder sieve in parallel vertical streams.* [b] *The radial lines emanating from the top view depict the characteristic decoders. The figure illustrates the case where* $(2^C + M)/M = 8$.

Figure 3: *Overall schematic of reduced hardware systolic super summation device.*

Figure 4: *Schematic of the sieve. Operands are bit-skewed: An summand's characteristic bits enter before its mantissa bits; high-order characteristic bits enter before low-order bits; high-order mantissa bits enter before low-order bits.*

Figure 5: *Each of the* 2^C *characteristic keys has* C *cells.*

Figure 6: *The cellular logic of mantissa bit movement through the sieve.*