Reducing Floating-Point Error Based on Residue-Preservation and Its Evaluation on an FPGA

Hasitha Muthumala Waidyasooriya, Hirokazu Takahashi, Yasuhiro Takei, Masanori Hariyama and Michitaka Kameyama
Graduate School of Information Sciences, Tohoku University
Aoba 6-6-05, Aramaki, Aoba, Sendai, Miyagi, 980-8579, Japan
Email: {hasitha, hirokazu, takei, hariyama, kameyama}@ecei.tohoku.ac.jp

Abstract—*Although scientific computing is gaining many attentions, calculations using computers always associated with arithmetic errors. Since computers have limited hardware resources, rounding is necessary. When using iterative computations, the rounding errors are added and propagated through the whole computation domain so that the final results can be completely wrong. In this paper, we propose a floating-point error reduction method and its hardware architecture for addition. The proposed method is based on preserving the residue coursed by rounding and reusing the preserved value in next iteration. The evaluation shows that the proposed method gives almost the same accuracy as the conventional double-precision floating point computation. Moreover, using the proposed method is 24% area efficient than using a conventional double-precision adder.*

Keywords: Precise arithmetic, floating-point, FPGA.

1. Introduction

Scientific computing is an area where mathematical models are executed in computers to analyze and simulate various physical behaviors. Such simulations are used in many fields such as fluid dynamics, molecular analysis and even in rocket science. Many of such models use repeated calculations spans many iterations. For example, finite-difference time-domain (FDTD) [1] used in fluid dynamics is such a well know method that deals with solving differential equations in a time-domain.

Although scientific computing is gaining many attentions due to the introduction of multicore CPUs and many core GPUs, calculations using computers are always associated with arithmetic errors. Due to the limited hardware resources in computers, rounding of the computation results is necessary. This gives a small error in many computations. Although such errors are negligible in a single calculation, they are a very big problem in scientific computing. The simulation models use repeated calculations with thousands of iterations to produce a result. Therefore, small error in each iteration add up and propagated through the whole computation domain. Due to this, the final results obtained after thousands of iterations might be completely wrong. Computation errors are been discussed in many works such as [2] and [3]. Accepting those results could bring devastating effects since many simulations are connected with real world application such as air plane designing, power plant controlling etc.

Easiest way of reducing computation error is to add more precision [4]. However, that comes with an increased hardware cost. Using software libraries such as "multiple precision integers and rationals (MPIR)" [5] is another way of dealing with this problem. However, when the precision increases the processing time also increases exponentially. In this paper, we focus on floating-point addition and propose a error-reduction method and its area-efficient hardware implementation. The proposed method based on a very simple idea of preserving the residue due to rounding and reuse it in recursive computation. We propose an efficient method implement this algorithm in smaller number of time steps. According to the evaluation using FPGA, the proposed single-precision floating-point adder gives almost the same accuracy of the double-precision floating-point adder, but requires 24% less area compared to the conventional double-precision adder.

2. Floating-point error reduction using residue-preservation

In this section, we focus on reducing the floating-point error due to normalization and rounding in iterative computations. In these computations, the output of the iteration i is used as an input of iteration $i+1$. Therefore, the error is propagated from iteration to iteration. However, if we can keep the residue of rounding in one iteration, we can use it in the next iteration. Even if the residue is very small during a single iteration, it will become large if we keep storing it. Therefore, after many iterations, the residue of rounding is also add up to the result and that will reduce the error. The algorithm to reduce the floating-point error in summation is given as follows.

Step 1: $R = S_0 = 0$
Step 2: $U = R + X_i$
Step 3: $S_{i+1} = S_i + U$
Step 4: $V = S_{i+1} - S_i$
Step 5: $R = U - V$

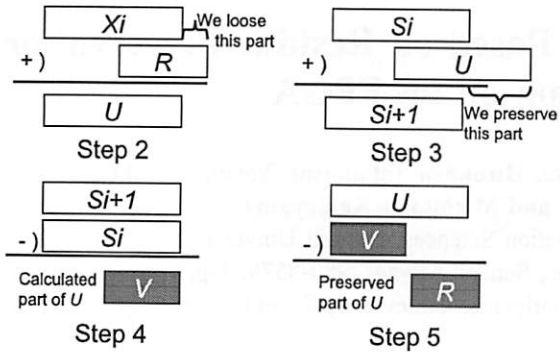


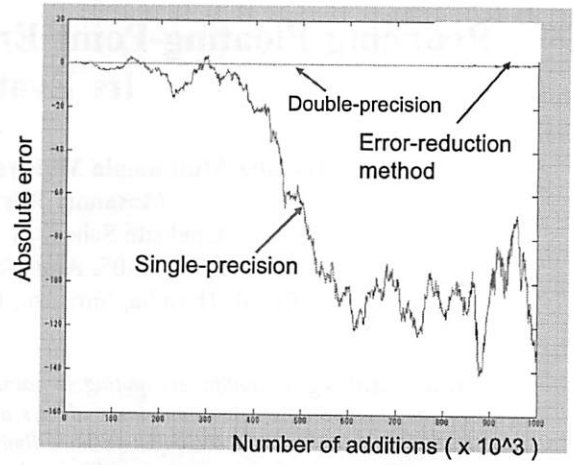
Fig. 1: Floating-point error reduction method

Step 6: if($i < n$), increase i by 1 and go to Step 2
else, finish

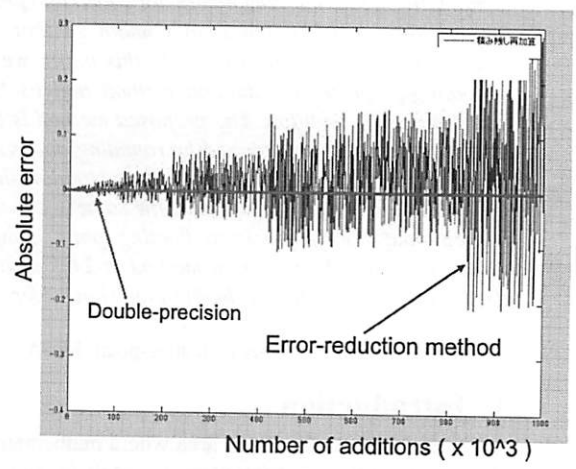
Figure 1 explains this algorithm using the computation of $\sum_{n=0}^{n-1} X_i$ as an example. In the first iteration, the value X_i is added with the residue R of the previous step. The result is saved as U as shown in Step 2. In this calculation, we loose a part of R due to rounding. Then we add U to the sum S_i to get the new summation S_{i+1}. Due to the rounding, only a part of U is added. This part V is found in Step 4. To find the non-added part of U, we subtract V from U in Step 4. Since this part is not added to the summation yet, we preserve it as R and use it in the next iteration.

Figure 2 shows the evaluation of this method. When the number of computations are large, this error reduction method with single precision computation gives extremely better results compared to conventional single-precision computation as shown in Fig.2(a). Moreover, the error reduction method gives very similar results to the conventional double precision computation. Note that, we calculate the error compared to the double-precision computation so that the error of doable-precision becomes zero. Figure 2(b) shows the graphs of the error reduction method and conventional double precision method to see the difference more clearly. There are two reasons for this difference. The first one is the rounding occurs in the conventional double precision computation. The second one is the unused residue occurs in the addition of X_i and R in Step 2 as shown in Fig.1.

Although this method gives a very good computation results, it has so many steps and need two additions and two subtractions. Therefore, if available, it is better to use a high-precision computation than using the error reduction method with low-precision computation. However, in the next section, we propose an improved algorithm combined with a new floating-point adder architecture to get the same error reduction under less additional computation and small hardware overhead.



(a) Computation error vs. number of additions



(b) Enlarged capture of Fig.2(a)

Fig. 2: Evaluation of the computation error

3. Proposed error reduction algorithm and its FPGA implementation

In the error-reduction algorithm explained in Section 2, the processing time is wasted in Steps 4 and 5 to calculate the residue occurs due to the rounding of S_{i+1}. However, if we can preserve all the bits of S_{i+1} before rounding, we can find the residue easily. This method is show as follows.

Step 1: $R = S_0 = 0$
Step 2: $U = R + X_i$
Step 3: $S_{i+1} = S_i + U$
$R =$ residue of rounded S_{i+1}
Step 4: if($i < n$), increase i by 1 and go to Step 2
else, finish

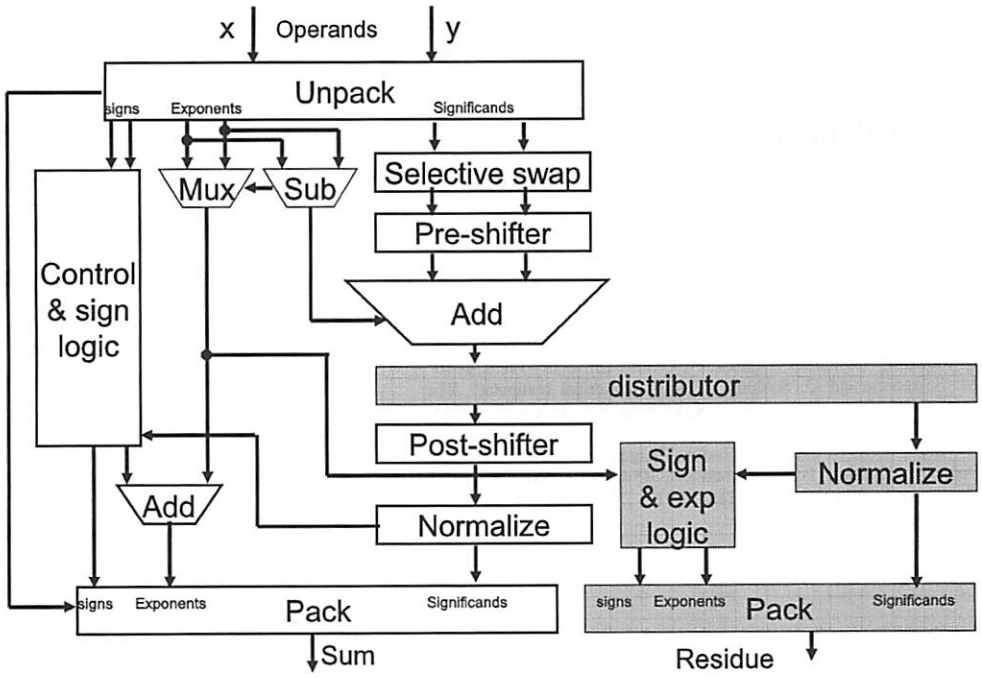


Fig. 3: Architecture of the proposed floating-point adder

Note that the residue calculation in Steps 4 and 5 are removed and the residue is preserved in Step 3.

To execute this algorithm, we proposes a new floating-point adder architecture as shown in Fig.3. The gray areas in Fig.3 shows the units we added to the conventional floating-point adder. To explain the architecture and the proposed algorithm, let us consider single-precision floating point addition. The "Add" unit shown in Fig,3 is the same one used in conventional single-precision adder. The only difference is that it produces two outputs; the normalized addition result and the residue after normalization and rounding. Since no extra adders are included, this architecture can be implemented area efficiently.

4. Evaluation

We implement the proposed floating-point adder on "Cyclone II EP2C35F6'2C6" FPGA to evaluate the error-reduction method. We used "Quartus II" software tool to calculate the number of logic elements (LEs) and the clock frequency. In the evaluation, the proposed method is compared with conventional single-precision and double-precision floating-point computations. Note that, we did not use any pipelines when implementing different adders. It is difficult to compare adders with different precisions with different pipeline stages.

Table 1 shows the evaluation results. According to the results, the proposed method requires less area than conventional double-precision floating-point method. However, the clock frequency is slightly lower than that of the double-precision method. As discussed in the previous section, the accuracy of the proposed method is much better than the single-precision and almost the same as the double-precision. Therefore, using the proposed method with single-precision is area-effective than using double-precision. However, as shown in 2(b) , if the number of iterations are extremely large as few millions, the difference between the proposed method and conventional double-precision method gets larger.

Table 1: FPGA evaluation of floating-point adders

	Conventional single-precision floating-point	Conventional double-precision floating-point	Proposed single-precision floating-point
Frequency	38 MHz	31 MHz	27 MHz
Num. LEs	611	1336	1014

5. Conclusion

We have proposed a floating-point error reduction method and its hardware architecture for addition. The proposed method based on preserving the residue coursed by rounding and reusing the preserved value for the calculation. The proposed adder store the residue in registers so that re-calculating of residue is not required. The evaluation shows that the proposed method gives almost the same accuracy as the double-precision floating point computation and more

area efficient than the double precision adder. In future works, we will extend the proposed method of other computations such as multiplication and division.

Acknowledgment

This work is supported by MEXT KAKENHI Grant Number 12020735.

References

[1] H. S. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media", IEEE Transactions on Antennas and Propagation, Vol.14, No.3, pp.302-307, 1966.
[2] B. Parhami, "Computer Arithmatic", Oxford University Press, 2010.
[3] M. Sofroniou and G. Spaletta, "Precise numerical computat", The Journal of Logic and Algebraic Programming, Vol.64, Issue 1, pp.113-134, 2005.
[4] Y. Hida , X. S. Li and D. H. Bailey, "Algorithms for Quad-Double Precision Floating Point Arithmetic", 15th IEEE Symposium on Computer Arithmetic, pp.155-162, 2001.
[5] http://www.mpir.org/