The 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
18-21 September, 2019, Metz, France

A Method of the Result Preparation in Addition-Based Circuits

Oleksandr Drozd[1], Anatoliy Sachenko[2], Konrad Grzeszczyk[3], Nadiya Vasylkiv[2],
Julia Drozd[1], Iryna Turchenko[2]
[1] Odessa National Polytechnic University, ave. Shevchenko, 1, Odessa, 65044, Ukraine,
drozd@ukr.net, yuliia.drozd@opu.ua
[2] Ternopil National Economic University, Peremoga Sq, 3, Ternopil, 46020, Ukraine,
as@tneu.edu.ua, nadiya.vasylkiv@gmail.com, itu@tneu.edu.ua
[3] International Vision Machinery, VISORT Sp., J., Radom, 26-600, Poland, info@visort.pl

I. INTRODUCTION

The method of preparation of results is widespread under various names and without them. It is enough to pay attention to various libraries, including IP-core library and library of the software, the prepared methods of its assessment .

The known component approach to creation of systems from the components taken “Off-The-Shelf” (OTS) including components of commercial use “Commercial OTS” or COTS components and components for critical applications “Critical OTS” or CrOTS components, is also implemented by method of preparation .

In the instrumentation and control safety-related systems ensuring functional safety of objects of the increased risk , scenarios and means of their realization for accident prevention and decrease in their consequences are prepared .

Fault tolerant decisions form a basis of functional safety, including the reconfigurable systems possessing a number of the prepared configurations , and the multi-version technologies preparing a set of versions of calculation performance in the solution of the same task for the purpose of opposition to common cause failure .

Besides, the prepared versions allow to raise a checkability of schemes and thus to reduce the risks connected with a problem of the hidden faults .

The speed of search engines is provided with preparation of results too .

The method of preparation of results allows to accelerate process of obtaining result by parallelization of processes of inquiry formation and preparation of a set of possible results.

An impression is originally made that the method of preparation of results is considerably labor-consuming. It is possible to assume that the method can be effective only if the set of the prepared results is used for the choice of rather large amount of the required results as it takes place in libraries. The case of the choice of one result seems unjustified from a position of complexity in realization of a method what imposes restrictions in our models concerning its application in circuitry for calculation of results when performing arithmetic operations.

However, application of a method of results preparation even for execution of basic arithmetic operation – addition which plays the main role when performing all other arithmetic operations is known.

In this case the method of results preparation carries out the accelerated propagation of carry in the parallel multidigit combinational fixed-point adder and is known as a method of carry precomputation in carry-selected circuitry. Its essence consists in splitting the parallel adder into identical sections which, since the second, are duplicated for two cases of receiving carry from an exit of the previous section: carry equal to zero, and carry equal to unit. All sections work at the same time, and the sums calculated by the sections duplicating each other are preparations. The choice of the required sum of result is carried out according to the calculated values of carry .

The method of carry-selected precomputation complicates the adder, but at the same time significantly reduces operation time: the linear dependence of time on size of operands can be reduced to the level of logarithmic dependence.

The method of carry-selected precomputation is the basis for creation of adders, comparators, counters and other similar arithmetical units in circuitry of the FPGA design which is carried out with the use of a modern CAD . Such schemes are under construction in the LUT-oriented architecture of FPGA with the use of the next logical LE elements which are adjusted on calculation of both sum and the carry of the full binary adder in the dynamic arithmetic mode. Carry, as well as the sum, is calculated for two values of an input signal of carry. Further both versions of a signal of carry propagate between the next logical LE elements by the prepared shortest ways .

FPGA of a chip are also preparations under projects, and results calculated on the programmed FPGA chip are prepared in memory of the LUT units which are a part of the logical LE elements .

Use of the method of precomputation in realization of basic arithmetic operation on modern technologies of FPGA design is an important argument in its advantage. Thanks to this method, arithmetic operations gain additional development in FPGA projects .

At the same time, according to resource approach, models, methods and means develop from simple to real on the way of structuring under features of the parallel and fuzzy natural world. Simple forms are initial exact and consecutive representations (models) and possibilities (methods) of the person. Real forms are parallel and approximate .

The dominating development in parallel processing of approximate data and the corresponding improvement of its hardware support in computer systems is shown on the example of personal computer evolution: from Intel 287/387 coprocessors of optional delivery to several floating-point pipelines in Pentium and several thousand such pipelines in the graphic processor used for performance of parallel calculations with the use of CUDA technology .

As a rule, approximate data are represented and processed in floating-point formats where the approximate nature of numbers is directly shown regarding mantissas .

Therefore, the purpose of this paper is distribution of a method of preparation of results on processing of approximate data on the example of floating-point addition.

Section II considers features in execution of floating-point addition. Section III describes a method of precomputation of results in relation to circuitry realization of floating-point addition of numbers. Section IV shows results of simulation of the floating-point adder designed on FPGA.

II. Features of Execution of Floating-Point Addition

Floating-point addition of two numbers presented as $A_{FP} = a_M\, 2^A$ and $B_{FP} = b_M\, 2^B$ where a_M and b_M – mantissas, A and B – exponents, is traditionally carried out by several consecutive steps .

The result $S_{FP} = A_{FP} + B_{FP}$ is represented as $S_{FP} = s_M\, 2^S$ by means of S exponent and s_M mantissa which are determined on steps 1 – 4 by the following formulas, respectively:

$$S = \max(A, B); \quad (1)$$

$$A^* = S - A,\ B^* = S - B; \quad (2)$$

$$a_{M\,SHIFT} = a_M\, 2^{-A^*};\ b_{M\,SHIFT} = b_M\, 2^{-B^*}; \quad (3)$$

$$s_M = a_{M\,SHIFT} + b_{M\,SHIFT}, \quad (4)$$

where A^* and B^* are the leveling differences determining the size of shift of mantissas;

$a_{M\,SHIFT}$ and $b_{M\,SHIFT}$ are the shifted mantissas.

Let the size of a mantissa and exponent be equal to n and r, respectively. Calculations are carried out on FPGA with the LUT-oriented architecture. The LUT unit is the generator of logic function of four variables and can work in the normal and arithmetic modes with a delay τ_N and τ_A, respectively .

Then time of calculation of the s_M mantissa can be estimated by the following formula:

$$T_A = T_{A1} + T_{A2} + T_{A3} + T_{A4},$$

where T_{A1}, T_{A2}, T_{A3}, T_{A4} are the delays in process of calculation of the s_M mantissa by steps 1 – 4, respectively.

Similarly, complexity of the calculation scheme of the s_M mantissa is beeing defined by quantity of the N_{A1}, N_{A2}, N_{A3}, N_{A4} LUT units used on each step 1 – 4: $N_A = N_{A1} + N_{A2} + N_{A3} + N_{A4}$.

The T_{A1} delay consists of comparison time of the exponents A, B and the choice time of the greatest of them. Comparison is most expedited by subtraction in the arithmetic mode for time $r\,\tau_A + \tau_N$ on $r + 1$ LUT units. The last LUT unit is used for carry transfer from a chain of its accelerated propagation to normal LUT exit. The greatest exponent is chosen for time τ_N by the parallel use of r LUT units. Thus, the step 1 is carried out for time $T_{A1} = r\,\tau_A + 2\tau_N$ with the use of $N_{A1} = 2\,r + 1$ LUT units.

Sizes of shift are calculated, starting with the younger bit, by the formula (2). The younger bit is formed for time τ_N after which the arithmetic shift along with calculation of the following bits of shift size begins to be carried out, according to formula (3).

Therefore, the step 2 brings $T_{A2} = \tau_N$ delay and it is carried out with the use of $N_{A2} = 2\,r$ LUT units.

Operation of shift is carried out on one-digit multiplexors, each of which is implemented in one LUT unit. The scheme of arithmetic shift executed on a step 3 contains r levels with n LUT units at each level. It determines $T_{A3} = r\,\tau_N$ and $N_{A3} = 2\,r\,n$.

The shifted mantissas are added on a step 4 with the use of $N_{A4} = n + 1$ LUT units for time $T_{A4} = n\,\tau_A + \tau_N$.

The total number of LUT units and operation time of 1 – 4 steps are determined as $N_A = 2\,r\,n + n + 4\,r + 2$ and $T_A = (n + r)\,\tau_A + (r + 4)\,\tau_N$, respectively.

The following steps in processing of a mantissa is determination of the direction and shift size of the s_M mantissa for normalization of result and performance of this operation.

Consecutive performance of all described steps leads to considerable addition time of mantissas. For reduction of duration of a clock cycle in a pipeline system, the sequence of steps can be divided into several sections of the pipeline.

However, in case of performance of the accumulating addition, such system will accumulate several sums, for example, the sums of numbers with odd and even numbers at division of the sequence of steps into 2 sections. Addition of these sums will face the same problem of consecutive performance of steps.

Therefore, there is a need for reduction of floating-point addition time. Such acceleration of calculations can be reached by application of a method of the results preparation.

III. Floating-Point Addition with Precomputation of Results

Method allows to carry out at the same time the processing of the exponents (steps 1 and 2) and addition of the shifted mantissas (a step 4). The step 3 is in fact the choice of operands for performance of operation (4). This step can be replaced with the choice of result (a step 5) from a set of the sums of mantissas with various situation from each other, i.e. with shift on one, two, …, $n - 1$ positions, where n is size of a mantissa.

For accounting of all mutual provisions of mantissas, it is necessary to prepare $2n + 1$ sums. In case of shift on n positions, the sum is not calculated as it is equal to one of a_M or b_M mantissas. The number of the calculated sums is equal to $2n - 1$.

The choice of each bit of $(n + 1)$-digit result is carried out from $2n + 1$ prepared sums.

We can estimate the expected complexity of the scheme in number of LUT units and time of calculations.

Steps 1 and 2 are carried out, as well as in the previous option, for time $T_{A1+2} = r\,\tau_A + 3\tau_N$ with the use of $N_{A1+2} = 4\,r + 1$ LUT units.

Step 4: Calculation of $2n - 1$ sums with the use of n-digit adders can be executed on $N_{A4.P} = (2n - 1)(n + 1)$ LUT units for time

$$T_{A4.P} = T_{A4} = n\,\tau_A + \tau_N.$$

The 5th step of choice of the prepared sums is carried out on multiplexors from two directions to one according to the pyramidal scheme of their connection. Each multiplexor is implemented by one LUT unit. The pyramidal scheme of the choice of one bit of result uses $2n$ LUT units. The step 5 is carried out for time $T_{A5.P} = (r + 1)\,\tau_N$ with the use of $N_{A5.P} = 2n\,(n + 1)$ LUT units.

The total number of LUT units and time of performance of 1, 2 and 4, 5 steps are determined as follows:

$$N_{A.P} = 4\,n^2 + n + 4\,r + 1;$$
$$T_{A.P} = \max(T_{A1+2}, T_{A4.P}) + T_{A5.P}.$$

In view of that $T_{A1+2} < T_{A4.P}$, as $r << n$, then $T_{A.P} = n\,\tau_A + (r + 2)\,\tau_N$.

The method of preparation of results reduces time of calculations by $\Delta T_{A.P} = r\,\tau_A + 2\,\tau_N$, when using $\Delta N_{A.P} = 2\,n\,(2\,n - r) - 1$ LUT units in addition. Considerable complication of the scheme limits to use of such decision.

The efficiency of offered method can be significantly increased in case of partial preparation of results.

We suggest to prepare the sums only for two cases: $A \geqslant B$ and $A < B$.

Step 1. Difference $A - B$: its value S_{A-B} and sign S_{SIGN} are calculated in the two's complement code. The S exponent is chosen from the exponent A and B under control of the S_{SIGN} sign.

Step 2. For mantissas b_M and a_M, two shifted mantissas b_{SHIFT} and a_{SHIFT} are prepared. For this purpose, the b_M mantissa is shifted arithmetically to the right (with loss of younger bits and filling of the released positions with the sign bit) on a size S_{A-B}. The a_M mantissa is shifted arithmetically to the right at a size

$$S_{B-A} = (\neg\, S_{A-B}) + 1. \qquad (5)$$

Performance of operations of inversion and incrementation in (5) does not demand complication of shift of the a_M mantissa in comparison with b_M mantissa. Inversion of bits of the size S_{A-B} at the address inputs of multiplexors is compensated by renumbering of their information bits upside-down as it is shown in Fig. 1 for the one-digit multiplexor.

Incrementation is considered by the shift of an initial mantissa on one position.

Step 3. Two sums $s_{M1} = a_M + b_{SHIFT}$ and $s_{M2} = b_M + a_{SHIFT}$ are prepared.

Step 4. The result of s_M gets out of two versions of s_{M1} and s_{M2}: $s_M = s_{M1}$ and $s_M = s_{M2}$ in case of $S_{SIGN} = 0$ and $S_{SIGN} = 1$, respectively.

Time and quantity of LUT units for performance of steps 1 – 4 is estimated by the following sums:

$$T_{A.PP} = T_{A1.PP} + T_{A2.PP} + T_{A3.PP} + T_{A4.PP},$$
$$N_{A.PP} = N_{A1.PP} + N_{A2.PP} + N_{A3.PP} + N_{A4.PP}.$$

The step 1 uses $N_{A1.PP} = 2\,r + 1$ LUT units, just as N_{A1}. The delay $T_{A1.PP} = \tau_N$ as the step 2 begins after calculation of the younger bit of shift size S_{A-B}.

The step 2 carries out in parallel two operations of arithmetic shift with delay $T_{A2.PP} = r\,\tau_N$ on $N_{A2.PP} = 2\,r\,n$ LUT units.

The step 3 executes in parallel two operations of addition with delay $T_{A3.PP} = n\,\tau_A + \tau_N$ on $N_{A3.PP} = n\,\tau_A + \tau_N$ LUT units.

The step 4 chooses result with delay $T_{A4.PP} = \tau_N$ with the use of $N_{A4.PP} = n$ LUT units.

The total number of LUT units and time of performance of 1 – 4 steps are determined as follows:

$$N_{A.PP} = 2\,r\,n + 3\,n + 2\,r + 3;$$
$$T_{A.PP} = n\,\tau_A + (r + 3)\,\tau_N.$$

The method with partial preparation of results reduces time of calculations on

$$\Delta T_{A.PP} = r\,\tau_A + \tau_N$$

when using $\Delta N_{A.PP} = 2\,(n - r) + 1$ LUT units in addition.

IV. Results of Simulation

Design of the floating-point adder regarding performance of steps 1 – 4 was carried out with the use of a CAD of Intel (Altera) Quartus II v. 13.0 SP1 Web Edition on a chip of FPGA Intel (Altera) Cyclone II EP2C35F672C6 for $n = 15$, $r = 4$ and $n = 31$, $r = 5$.

Calculated estimates show complication of the floating-point adder scheme by 15.0% and 14.6% in case of $n = 15$ and $n = 31$, respectively. Results of design determine estimates, close to that. The scheme becomes complicated by 16.0% and 14.9%.

Calculated values of T_A and $T_{A.PP}$ and also results of simulation of operation time $T_{A.E}$ and $T_{A.PP.E}$ with the use of the TimeQuest Timing Analyzer utility for the initial scheme of the adder and scheme with the partial preparation of results are shown in table 2.

Calculated estimates show decrease of operation time from $T_A = 9.06$ ns to $T_{A.PP} = 6.83$ ns and from $T_A = 10.43$ ns to $T_{A.PP} = 8.10$ ns, i.e. by 24.6% and 22.2% for $n = 15$ and $n = 31$, respectively.

Results of simulation show bigger time in comparison with calculations, but also confirm advantages of the offered method in time which decreases from $T_{A.E} = 9.76$ ns to $T_{A.PP.E} = 7.33$ ns and from $T_{A.E} = 10.72$ ns to $T_{A.PP.E} = 8.87$ ns, that is by 24.9% and 17.3%.

Conclusions

In this paper the method of preparation or precomputation of results which reflects trends of development of computer engineering is considered.

Possibilities of this method are shown on reduction of operation time in FPGA implementation of the floating-point adder.

Further researches are planned in the direction of development of the results preparation method for other arithmetic operations and their on-line testing taking into account truncated execution of calculations and their residue checking .

The prospects of such researches are based that the arithmetic shift belongs to the truncated operations. Besides, operands and results in on-line testing of approximate calculations are mantissas and their short check codes.

It promotes increase in efficiency of precomputation method which replaces operation of the choice of operands with simpler operation of the choice of results.