Sharing of Functional and Special Means in Pipeline Floating-Point Systems with Strongly Connected Versions

Oleksandr Drozd[1], Igor Kovalev[1], Myroslav Drozd[1], Oleksandr Martynyuk[1], Serhii Polozhaenko[1]
[1] Odessa National Polytechnic University, ave. Shevchenko, 1, Odessa, 65044, Ukraine,
drozd@ukr.net, igoryan33@ua.fm, myroslav.drozd@opu.ua, anmartynyuk@ukr.net, sanp277@gmail.com

I. INTRODUCTION

The offered research lies on crossing of two important directions in development of computer systems:

- the multi-version technologies which are aimed at increase in functional safety of critical applications;
- processing of approximate data in the floating-point formats reflecting a tendency to growth of the importance of approximate calculations, in particular, in processing of measurement results which are initial data in the safety-related systems.

Multi-version technologies belong to advanced fault tolerance decisions. They play an important role in ensuring functional safety of the instrumentation and control safety-related systems and their objects of control which are characterized by the increased risk of operation.

Powerful power supply systems belong to them: power plants and power grids and also aircraft and land high-speed transport, different types of arms and spacecrafts.

Multi-version systems offer several versions of the solution of the same task, for example, a majority voting system which channels are developed with the use of various types of a diversity aimed at providing various reaction of versions to the same failures. Channels can be developed by various teams with the use of various tools. Such approach allows to detect and parry common cause failures, including design errors and also to reduce a set of the hidden faults which can be accumulated in normal mode of safety-related systems and be shown in its emergency operation.

It should be noted that the initial majority voting system with three identical channels already is fault tolerant in relation to fault in the scheme of one channel. This fault tolerance is reached thanks to the natural version redundancy based on natural types of a diversity. Channels cannot be made absolutely identical. They are already versions, and their natural version redundancy is sufficient for obtaining the fault tolerant decision. The requirement to resist to common cause failures is determined by the international standards which regulate norms for ensuring functional safety of instrumentation and control safety-related systems and objects of the increased risk.

Problem of multi-version systems is their complexity which grows together with the number of versions and therefore limits development of these systems. However, such complication of the multi-version systems is not their integral property, as well as full mutual independence of versions. In the natural world multi-version systems show the inverse property: versions are functionally connected among themselves.

For example, fingers of a hand are versions of each other, but is together functional more richly than each of them. FPGA design raises a checkability of the scheme and trustworthiness of results when using several connected versions of the program code.

In traditional multi-version systems, the possibility of parrying the common cause failures follows from maximum mutual independence of all versions i.e. in each their couple. Independent versions are not dependent on the same fault. In other words, if versions are independent, then the fault caused by the common reason is parried. Such definition of the problem narrows a number of decisions as the purpose to parry this fault is changed to the purpose of obtaining independent versions.

In fact, it is important to estimate as far as versions can be connected among themselves for parrying of common cause failures. Such approach defines systems with strongly connected versions which refuse full independence of versions and allow their general parts, for example, in residue number systems with detection and correction of errors.

Such systems contain k blocks. Each of them executes processing of numbers on their remainders of division on one modulo of k of mutually simple modules. Any subset from $k - 2$ blocks provides functionality of a system, i.e. is its version. Two more blocks allow to have two right versions in case of refusal of any one block. These versions allow to find and exclude the faulty block from calculations.

Parrying of common cause failures in the connected versions is based on two conditions:

- the set of versions has to contain at least one right version, i.e. the version which is independent of fault;
- there has to be an acceptable method of finding of the right version.

The connected versions together have considerably smaller complexity in comparison with a set of the same, but independent versions.

Modern computing systems are developed like pipeline. Their sections are the arithmetic units constructed on the basis of the matrix structures made of uniform operational elements with regular connections. Examples of such arithmetic units are parallel adders and shifters, iterative array multipliers and dividers which are close to the organization of systems with strongly connected versions.

The safety-related systems including the cyber-physical and the Internet of things systems relating to critical applications process the results of measurements received from sensors. Processing of these approximate data is carried out in floating-point formats where numbers are represented with a mantissa and an exponent.

Interaction between them is carried out by means of operations of a denormalization of operands and normalization of the results, as a rule, applied when performing each floating-point operation. Operations of normalization and a denormalization are executed with the use of an arithmetic shifter of mantissa code.

One of the easiest ways of search of the right version consists in consecutive change of versions before emergence of the required version. Identification of the right version is carried out by control of result with the use of on-line testing methods.

In matrix structures of the arithmetic units transformed to the organization of systems with strongly connected versions, change of versions is carried out with the use of cyclic shifters which position codes of operands and choose codes of result.

Shifters of binary codes in circuits with matrix structures are rather complex arithmetical units which make a considerable part of expenses of the equipment both in systems with strongly connected versions and in pipeline floating-point systems.

Therefore, we offer researches on sharing of shifters in these systems. The main scientific contribution consists in combining special means of the choice of the right version and functional means of processing of approximate data in the systems of critical application.

Section 2 considers features of systems with strongly connected versions in relation to matrix structures of arithmetic blocks on the example of the iterative array multiplier. Section 3 analyzes possibilities of joint realization of cyclic and arithmetic shifter for the choice of the right version in the course of a denormalization of operands. Section 4 represents results of experiments on combination of cyclic and arithmetic shifter at their design on FPGA.

II. Features of Systems with Strongly Connected Versions

We will understand as the version such smallest part of a system with strongly connected versions which can replace a system in the absence of faults. Addition of the version is the rest of a system, independent of it.

The right version of a system exists at its one fault if additions of all versions are mutually independent and cover all system.

Really in this case, fault of a system belongs to one addition which version is true.

We will consider the organization of a system with strongly connected versions on the example of the iterative array multiplier which carries out multiplication of n-bit binary codes for one clock cycle and contains n of identical lines of operational elements for multiplication of bits of a multiplier by the code of a multiplicand.

The matrix of the multiplier is complemented in one more line which connects to the last and first line, forming ring structure with an excess line. This structure is complemented with elements of a rupture of a ring after each next line, defining the next line superfluous. The excess line is addition for the version consisting of n of the previous lines. The system excludes a faulty line from computation process if this line becomes superfluous.

The system contains a set of $n + 1$ versions which are in pairs connected by the general $n – 1$ lines. Their general part $\Delta = 1 – 1 / n$ increases with growth of n, for example, $\Delta = 87.5\%$ and $\Delta = 98.4\%$ for $n = 8$ and $n = 64$, respectively. At the same time, relative complication of a matrix regarding operational elements decreases on $\delta = 1/n$ part: $\delta = 12.5\%$ and $\delta = 1.56\%$.

Independence of additions, i.e. lines of a matrix, it is necessary to provide regarding the duplicated multiplicand bits which create ways for propagation of an error between lines.

For this purpose, inversions of these bits come to inverse inputs of two AND elements consistently connected. The first AND element executes logical multiplication of the bit of a multiplier and the main bit of a multiplicand. The second AND element calculates conjunction of the received result on the duplicating bit of a multiplicand. Fault of one of the duplicating inverse bits of a multiplicand can extend to other lines only at zero value. In this case in correct lines, one AND element becomes the repeater. Other AND element calculates the correct conjunction of the bit of a multiplicand on the bit of a multiplier.

It should be noted that the younger part of a matrix of conjunction of the product excluded from calculations in a method of truncated multiplication shows faults in the form of the errors which are inessential for trustworthiness of result.

When changing versions, the most part of a matrix estimated as $\Delta_M = (n - \log_2 n) / n$ appears on the place of its younger part: $\Delta_M = 84.4\%$ and $\Delta_M = 90.6\%$ for $n = 32$ and $n = 64$, respectively.

On-line testing methods distinguishing essential and inessential errors, allow to mask the faults which are not breaking trustworthiness of result.

The main complication of the multiplier in a system with strongly connected versions is associated with transition to the following version which is provided with cyclic shift.

III. Joint Realization of Cyclic and Arithmetic Shifters

According to standards of a floating-point arithmetic, the M mantissa of the normalized number changes in all basic and extended binary formats in the limits determined by inequality: $1 \le M < 2$.

This inequality determines change ranges for mantissas of results in all operations with the normalized numbers.

Operation of multiplication of two normalized mantissas defines a M_P product mantissa in limits: $1 \leqslant M_P < 4$. Normalization of result regarding the M_P mantissa is carried out by its shift on one position to the right with loss of the younger bit in case $M_P \geqslant 2$.

Operation of division of the normalized mantissas defines a M_Q quotient mantissa in limits: $0.5 \leqslant M_Q < 2$. Normalization of result regarding a mantissa of M_Q is carried out by its shift on one position to the left with filling of the younger bit with zero or value of the sign in case $M_Q < 1$.

Operation of addition or comparison of the normalized mantissas is carried out by alignment the exponents. If exponents differ on L, then denormalization of the mantissa with a smaller exponent by shift on L positions to the right with loss of the L younger bits is carried out. The L senior bits accept zero value or value of the sign.

The M_A mantissa of addition result can be calculated in limits: $0 \leqslant M_A < 4$. In case of $M_A \geqslant 2$, normalization is carried out like result of multiplication, and by cyclic shift at $0 \leqslant M_A < 1$.

We will consider joint performance of arithmetic and cyclic shift respectively on L and C positions to the right for the most difficult case of a denormalization of a mantissa in operation of a floating-point addition.

Joint performance of arithmetic and cyclic shift of a n-bit $M\{1, \ldots, n\}$ mantissa where the bit $M\{1\}$ is a senior, is implemented in several steps.

Step 1. Auxiliary n-bit code $H\{1, \ldots, n\}$ containing zero and unit respectively in L younger and $n - L$ senior bits is formed. The bit $H\{1\}$ is a senior. The i bit, $i = 1, \ldots, n$, of this code accepts the value calculated by the following formula: $H_i = (i > L)$.

Then the code is divided into two parts:

$$H\{1, \ldots, n - L\} = 2^L - 1;$$
$$H\{n - L + 1, \ldots, n\} = 0.$$

Step 2. The $M_0\{1, \ldots, n\}$ code of mantissa with zero on the place of the younger lost bits is calculated with the use of the AND operation for each bit by the following formula:

$$M_0\{1, \ldots, n\} = M\{1, \ldots, n\} \text{ AND } H\{1, \ldots, n\}.$$

In this case, the code is divided into two parts which are determined as follows:

$$M_0\{1, \ldots, n - L\} = M\{1, \ldots, n - L\};$$
$$M_0\{n - L + 1, \ldots, n\} = 0.$$

In case of filling of the released positions with value of the sign in the one's or two's complement code, the $M_S\{1, \ldots, n\}$ code of mantissa is calculated by the following formula:

$$M_S\{1, \ldots, n\} = M_0\{1, \ldots, n\} \text{ OR } S \text{ AND } \neg H\{1, \ldots, n\},$$

where $S = S\{1, \ldots, n\}$ – the n-bit code made of values of the sign.

The step 3 is carried out along with steps 1 and 2 and consist in calculation of the following sum:

$$S_{L+C} = (L + C) \bmod n,$$

where it is necessary to subtract n in case of $L + C > n$.

Step 4. The code of result of the joint shift operation is calculated by cyclic shift of $M_0\{1, \ldots, n\}$ or the $M_S\{1, \ldots, n\}$ code, calculated on the step 2, on S_{L+C} positions to the right.

We can estimate complexity of performance of the joint shift operation on FPGA with the LUT-oriented architecture.

Let LUT generate function of four arguments.

Steps 1 and 2 are performed by implementation of functions no more than $R + 1$ or $R + 2$ arguments for calculation of each bit of $M_0\{1, \ldots, n\}$ or the $M_S\{1, \ldots, n\}$ code, respectively, where R is a size of the L and C codes, $R =]\log_2 n[$. These arguments are no more than R bits of the C code, the bit of the $M\{1, \ldots, n\}$ code and the bit of the sign in case of calculation of the $M_S\{1, \ldots, n\}$ code.

We can expect use of n LUT units for calculation of $M_0\{1, \ldots, n\}$ code for $R \leqslant 3$ and the $M_S\{1, \ldots, n\}$ code for $R \leqslant 2$ and also no more $2n$ LUT units for calculation of $M_0\{1, \ldots, n\}$ or $M_S\{1, \ldots, n\}$ code in case of $3 < R \leqslant 7$ or $2 < R \leqslant 6$, respectively.

The complexity of performance of the step 3 can be estimated as $2R$ LUT units.

The cyclic shifter which is carrying out the step 4 completely coincides with an initial cyclic shifter on C positions and coincides in complexity with an arithmetic shifter on L positions to the right. The cyclic shifter constructed of R levels with n multiplexors from two directions to one in each level is implemented with the use of $R\,n$ LUT units.

Thus, joint performance of arithmetic and cyclic shift can be executed with the use of 104 and 524 LUT units for $n = 16$ and $n = 64$, respectively. The total complexity of two initial shifters is 128 and 768 LUT units. Schemes become simpler for 23.1% and 46.6%.

IV. FPGA Design of Original Block

The scheme of joint performance of arithmetic and cyclic shift contains the original block which is carrying out steps 1 and 2, and standard blocks of the adder and a cyclic shifter.

For the experimental evaluation of the suggested method in the Quartus Prime 18.1 Lite Edition CAD system, the VHDL-projects of original block for $n = 7$, 15 and 31 have been implemented on Intel Max 10 FPGA 10M50DAF672I7G.

The designed scheme of original block for $n = 7$ and $R = 3$ is shown in Fig. 1.

This scheme contains 3-bit control input L{1, …, 3}, 7-bit Inpf{1, …, 7} input of 7-bit operand, 7-bit Outf{1, …, 7} of 7-bit result output and $n = 7$ LUT units according to condition $R \leqslant 3$.

In case of $n = 15$ and $n = 31$, the circuit consists of 23 and 55 LUT units accordingly. The experiment confirms estimates of complexity of schemes regarding the proposed solution.

Conclusions

In this work, we offered a method of joint performance of arithmetic and cyclic shift. This method allows to combine special means of search of the right version in the multi-version systems with strongly connected versions and the functional means of interaction of a mantissa and an exponent in pipeline floating-point systems.

Such combination is useful, considering the interrelation existing between multi-version systems and pipeline floating-point systems. This communication consists in the general the field of use, important for them, as components of the safety-related systems. Multi-version components resolve issues of functional safety of a system and an object of the increased risk. Pipeline floating-point systems are a basis for realization the multi-version components as they carry out pipeline processing of the approximate data obtained from sensors of the safety-related systems.

Cyclic and arithmetic shifters make a considerable part of the multi-version systems with strongly connected versions and pipeline floating point systems. Their joint realization leads a component of critical application to significant simplification.

The offered method adds a synergy factor to uniform realization of these systems.