Speed-Independent Floating Point Coprocessor*

Stepchenkov Y.A., Zakharov V.N., Rogdestvenski Y.V., Diachenko Y.G., Morozov N.V., Stepchenkov D.Y.
Department of perspective computer systems architecture
Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences (IPI FRC CSC RAS), IPI RAS, Moscow, Russian Federation
{YStepchenkov, VZakharov, YRogdest, YDiachenko, NMorozov, DStepchenkov}@ipiran.ru

Abstract

Speed-independent fused multiply-add unit as a coprocessor is represented. It purely conforms to IEEE 754 Standard. For minimization hardware and power consumption, a number of pipeline stages is reduced down to two. Wallace tree in the multiplier utilizes redundant self-timed code. Represented unit is developed on a base of standard 65-nm CMOS bulk process. It provides a performance up to 0.54 Gflops, and power consumption at level of 450 mW/Gflops.

1. Introduction

Fused Multiply-Add (FMA) is a standard operation in the modern computers. Most publications cover synchronous FMA units [1]. But during the last years, many publications describing asynchronous FMA unit implementations have appeared [2]. The latest solutions based on weak transistors do not meet the need to develop jam-resistant and energy-effective ST-units with proper operation not depending on element's delay, i.e. Speed-Independent (SI) circuits.

SI-circuits provide reducing power consumption due to removing both clock generator, and "clock tree" out of a circuit. This allows cutting down power consumption by 30 ÷ 50%. SI-basis ensures switching to the energy-conscious mode of operation of the hardware parts not used at the current cycle of data processing (for example, by dramatic lowering power supply for these parts). SI-circuits consuming very small power open wide perspectives for energy-efficient hardware development. Hardware redundancy and additional delays for indication, which are the intrinsic features of the SI-circuits, are the payment for such advantages.

However, a proper designing of SI-circuits allows reducing this redundancy essentially, and, in some cases, even to achieve better results compared to the synchronous analogs [3].

The purpose of investigation is to design 64-bit SI floating point coprocessor (SIFPC) of Gflops range utilizing FMA operation and conforming to IEEE 754 Standard. Such unit provides a faultless data processing in wide range of supply voltage and temperature. This is an actual problem for stream calculations, as well as for constructing modern super-computers.

2. Features of SIFPC

SIFPC's field of application (modern computing aids) puts in the forefront minimal power consumption at sufficiently high performance as the main requirement. This is caused by rather low clock frequency as well as by large number of SIFPC units per each VLSI for high performance computers. The analysis of minimax curve drawn at the axes of power consumption and die size in accordance with the technique in [4] made for 65-nm standard CMOS process and expected performance has allowed to determine the major prototype's characteristics, as well as to choose the structure chart for its implementation (Figure 1).

Inputs of SIFPC are processed operands (*X*, *Y*, *Z*), the attributes of treated operation (*R*) and initial reset (Rst). The results are sum (*FMA*) and operation flags (*RFA*).

RqI and *AckI* signals provide an input asynchronous interface: first reflects input data availability, while the other validates successful data acquiring termination by SIFPC. Similarly *RqO* and *AckO* signals provide an output asynchronous interface: *RqO* indicates result availability; *AckO* validates that asynchronous environment has terminated reading this result.

The most effective multiply algorithms are based on

* The reported study was funded by RFBR according to the research project (№ 13-07-12068 ofi_m).

978-1-4673-7776-8/15/$31.00 ©2015 IEEE

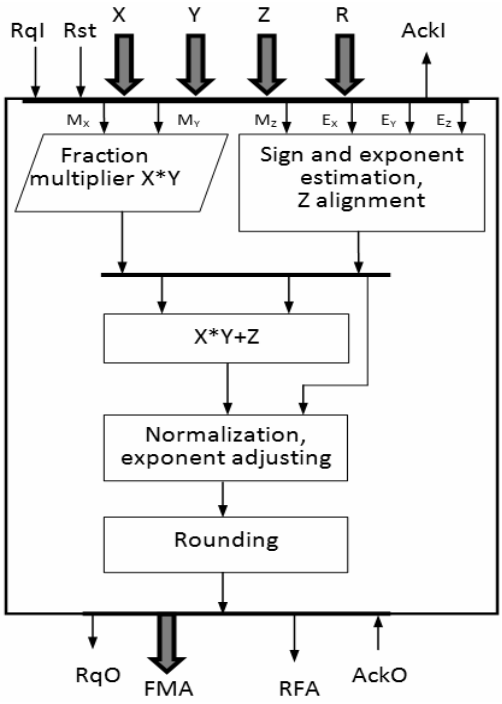

Figure 1. Structure chart of SIFPC

Booth coding algorithm modifications and on pipelined Wallace tree addition. The last causes major delays and hardware costs. The current trend in the Wallace tree algorithms intended for the modern multicore processors is either to minimize a number of pipeline stages, or not to use them at all. SIFPC meets this trend implementing fraction multiplication, result's exponent calculation, and third operand alignment as one stage of the total pipeline.

To minimize hardware and power consumption, it is reasonable to implement entire SIFPC as a single stage combinational unit. But such approach leads to its low performance due to lots of sequential operations.

The circuit in the Figure 1 illustrates a reasonable structure chart of pipeline for dual-rail data coding which is typical for SI-units. However, the researches have shown that a redundant (ternary) self-timed (ST) coding provides the best parameters of the SIFPC.

2.1. Redundant ST-coding

Multiplier is the most complex unit in SIFPC. So we focused on its optimal circuitry. A synchronous Wallace tree implementation [5] was selected as a prototype. It utilizes a redundant coding for operand representation providing fourfold degree of reduction at first stage and double degree of reduction at all following stages. Analysis of possible ST-codes has led to redundant ST-code presented in Table 1.

Table 1. Redundant ST-code

Coded state	Redundant code		
	Ap	*Am*	*An*
+1	1	0	0
0	0	0	1
–1	0	1	0
spacer	0	0	0

Figure 2 demonstrates SI-circuit of one bit of a SI ternary adder with redundant inputs and outputs. Indication subcircuit indicating outputs of all cells within this circuit occupies right side of the Figure 2. Redundant ST-coding improves Wallace tree implementation. It increases the multiplier's performance by more than 20%

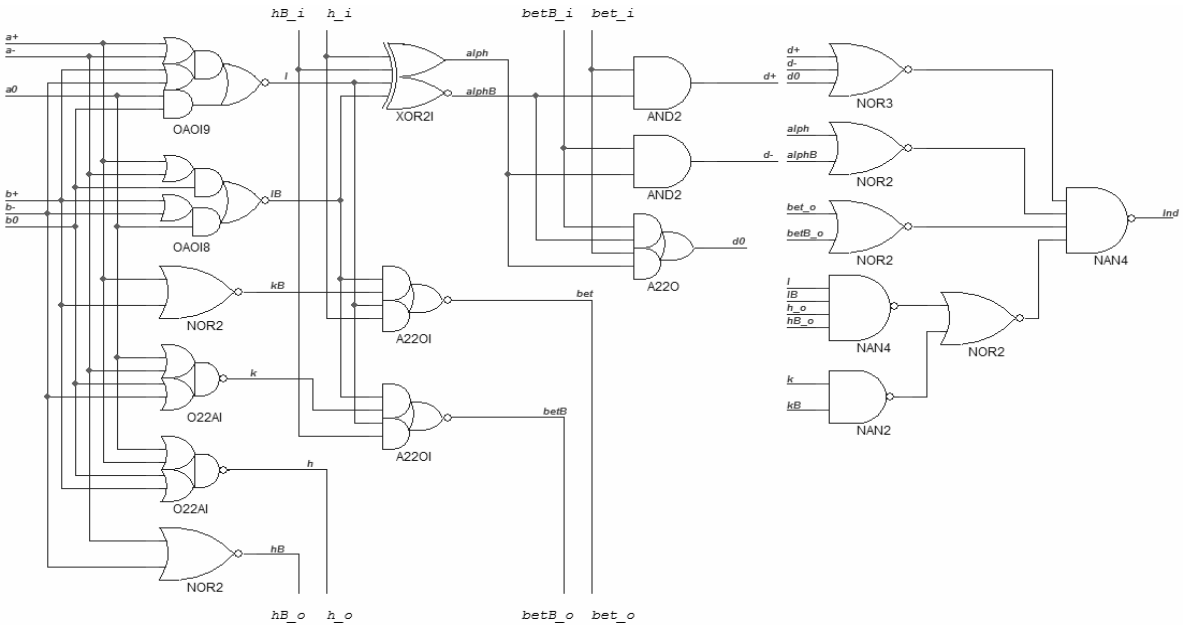

Figure 2. Bit of SI ternary adder

in comparison with traditional ST-tree with dual-rail coding. In addition, it reduces hardware by 20%, due to the number of compression stages drops from 7 to 4 stages.

Multibit adders with redundant ST-coding are free of ripple-through carry. This essentially speeds up their performance. So it is reasonable to use the redundant ST-coding not only in multiplier, but also on the following data processing stages.

Maximum performance of SIFPC is achieved due to the pipeline architecture as in synchronous circuits. However, the redundant ST-coding changes the rations between the times of the multiply and add-subtract operations. In turn, this causes to look at optimal pipeline structure in a new fashion.

2.2. SIFPC's pipeline

A pipeline of SI-unit is traditionally based on request-acknowledge interaction between its stages. Each pipeline stage contains an input or an output register storing intermediate data. The number of the information signals in the SI-circuits manifold exceeds the number of signals in synchronous analogs. Dual-rail coding doubles the number of signals. Therefore the registers in SIFPC pipeline stages have a large width, and lead to an essential complication of the hardware constructed in accordance with Figure 1.

Minimization of the SIFPC pipeline stage number, i.e. merging all data processing algorithms following multiplier into one pipeline stage, allows reducing amount of the registers with their indication subcircuits. This simplifies hardware, decreases its power consumption, and provides a parity of the times needed for both stages to switch into work and spacer phase.

The main difference between SIFPC stage and typical one consists in usage of the input register on a base of the hysteresis triggers [6] (H-triggers). They work in the same manner as well known C-element does. But they do not include "weak" transistors, and due to this have higher noise immunity. Figure 3 illustrates the one bit of such register. Here *X*, *XB* is dual-rail data input with null spacer; *E* acts as a common control signal for entire register generated by indication outputs of this stage and the following one; *Ii* is a bit-wise indicator of a combinational part of the previous stage; *Io* is indication output; *Y*, *YB* is dual-rail data output with null spacer.

Such register keeps both a work phase, and a spacer phase of the input *X*, *XB*. This allows using none additional unit at the input of each SIFPC pipeline stage converting bi-phase signals formed by traditional storage register, into dual-rail signal needed by combinational SI-circuits. Besides, such register indicates the bit-wise indicators of combinational part of the previous stage. As a result, an indication subcircuit becomes simpler and faster. Resulted interaction scheme for SIFPC pipeline stages fully responds to the traditional handshake scheme for SI-units [6].

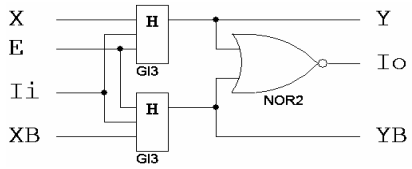

Figure 3. Input register bit

Described organization of the request-acknowledge interaction between SIFPC pipeline stages was successfully verified by means of ASPECT program [7] analyzing SI-features of any circuit.

2.3. SIFPC indication solutions

An indication of termination of all transients in a circuit in each phase of its work ensures its correct functioning as SI-circuit. However, indication subcircuit is a bottleneck of any multibit SI-unit essentially decelerating it.

A "classic" SI-implementation requires indicating all cells of the circuits in each phase of its work.The offered approach to SIFPC realization features usage of necessary and sufficient, but simplified indication of a work phase in each pipeline stage. Taking into account that intput register in each stage stores and indicates data processing results obtained by the previous stage, indication of combinational part of the current stage in work phase is not required. Using dual-rail and redundant ST-codes guarantees a single switch of activated elements of combinational part during transition of entire pipeline stage from spacer to a work phase. So appearance of the work state after spacer at the information outputs in all bits of the input register guarantees a readiness of a result.

On the contrary, at switching combinational part to spacer, all circuit elements are indicated, as we must make sure that all of them have terminated their initiated transitions and went into spacer state before allowing circuit to switch to the next work state. Otherwise hazards and bounces are possible at the outputs of elements, which is a violation of the SI-realization principles.

Simplified indication can lead to self-timed fault only in a case, when circuit contains a cell with very large transport delay of transition from spacer to work phase, which is able to cause belated switching of the cell into work state already after supplying next spacer to its inputs. If this delay will exceed a total duration of the work and spacer phases, then such cell will switch into work state at consecutive work phase, not at current phase. This may cause hazards during

switching circuit into an ordinary work phase.

Such simplified indication reduces complexity of SIFPC indication subcircuit implementation by 40-50% and accelerates it by 20-30%. For example, one bit of the Wallace tree in Figure 2 with simplified indication demonstrates higher by 50.9% performance, and less by 70% complexity of the indication part in CMOS transistors comparing to variant with full indication.

3. SIFPC parameters

SIFPC was developed in standard 65-nm CMOS bulk process with 6 metal layers. Table 2 summarizes its parameters. Time and energy parameters have been obtained by simulation with parasitic capacitances and resistors extracted from the layout for statistically reliable set of input operand triplets.

Table 2. SIFPC's parameters

Parameter	Value
Complexity, transistors	315 000
Die size, mm^2	0.47
Performance, Gflops	0.54
Latency, ns	1.9
Power consumption, mW/Gflops	450

Performance was calculated for typical operation conditions (1.0 V power supply, 25 Celsius degree), as a speed of SI-circuits always corresponds to the current environment conditions, and SI-circuits do not require taking into account the worst case.

Thus, usage of SIFPC in modern computing systems and nets is reasonable. As single, it provides performance at 0.54 Gflops level. However, modern computing aids achieve high performance, first of all, due to paralleling calculations and using greater amount of the processors, but not due to advancing performance of each processor. To get higher performance, one should use two and more parallel SIFPC units.

This also helps to resolve a problem of advancing reliability of the modern computing systems, for example, by means of doubling SIFPC units that are purely self-checking with respect to constant failures, for obtaining their fail-safe implementations.

The investigations show [8] that taking into account a character of processed operands allows to accelerate SI computers essentially and simultaneously reduce their power consumption. So following work will be directed on implementation of optimized architecture of the SIFPC providing a dynamic exclusion of some blocks and pipeline stages from SIFPC, which are not needed for processing the current input operands.

4. Conclusions

The usage of SI-circuitry for implementing modern computing systems helps to utilize the hardware methods for controlling reliability and validity of calculation results efficiently.

The scientific novelty consists in usage of the redundant ST-coding, simplified indication and minimal pipeline stage number. These have provided designing the competitive by performance 64-bit coprocessor implementing FMA operation and possessing all advantages of the SI-units: pure self-checking with respect to constant failures, retention workability at very-small supply voltage.

Practice value of the investigation is proved by developed 65-nm CMOS SIFPC demonstrating the high average real performance (0.54 Gflops at typical conditions), low latency (no more than 1.9 ns), and low power consumption (450 mW/Gflops).

5. References

[1] P.-M. Seidel, "Multiple Path IEEE Floating-Point Fused Multiply-Add", *46th IEEE International Midwest Symposium on Circuits and Systems*, Cairo, Egypt, 2003, pp. 1359–1362.

[2] R. Manohar, and B.R. Sheikh, *Operand-Optimized Asynchronous Floating-Point Units and Method of Use Therefor*, US patent, № 20130124592. May 2013.

[3] Y. Stepchenkov, Y. Diachenko, V. Zakharov, Y. Rogdestvenski, N. Morozov, and D. Stepchenkov, "Quasi-Delay-Insensitive Computing Device: Methodological Aspects and Practical Implementation", *International Workshop on power and timing modeling, optimization and simulation*, Delft, Netherlands, 2009, pp. 276–285.

[4] S. Galal, and M. Horowitz, "Energy-Efficient Floating-Point Unit Design", *IEEE Transactions on computers*, 2011, V.60, No.7, pp. 913–922.

[5] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Mashiko, "An 8.8-ns 54x54-bit Multiplier With High Speed Redundant Binary Architecture", *IEEE Journal of Solid-State Circuits*, 1996, V.31, No.6, pp. 773-783.

[6] Varshavsky, V., M. Kishinevsky, V. Marakhovsky, et al. *Self-timed Control of Concurrent Processes*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1990.

[7] Y. V. Rozhdestvenskii, N. V. Morozov, and A. Rozhdestvenskene, "ASPECT: A Subsystem for Event Analysis of Self-Timed Circuits", *Perspective micro- and nanoelectronics systems development problems*, Moscow, 2010, pp. 26–31 (in Russian).

[8] B.R. Sheikh, and R. Manohar, "Operand-Optimized Asynchronous IEEE 754 Double-Precision Floating-Point Adder", *IEEE International Symposium on Asynchronous Circuits and Systems*, 2010, pp. 151-162.