Energy Efficient Speed-Independent 64-bit Fused Multiply-Add Unit*

Yury Stepchenkov, Dmitry Stepchenkov, Yury Rogdestvenski, Yury Shikunov, Yury Diachenko
Institute of Informatics Problems
Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences
Moscow, Russia
YStepchenkov@ipiran.ru, stepchenkov@mail.ru, yrogdest@ipiran.ru, yishikunov@gmail.com, diaura@mail.ru

***Abstract* — The results of a Speed-Independent Fused Multiply-Add (SIFMA) unit pipeline implementation research are presented. SIFMA is compliant with IEEE 754 Standard. A criterion of the SIFMA pipeline's maximum performance is formulated. A method of reducing hardware costs of SIFMA multiplier by 1.5-2 times depending on its features is offered. The multiplier utilizes a modified Booth algorithm using self-timed redundant code. A new energy efficient self-timed organization of an input and output FIFO was developed. It provides less complexity versus a previous SIFMA implementation on base of semi-dense register.**

Keywords— self-timed circuit; multiplier; adder; subtractor; pipeline; indication, FIFO

I. INTRODUCTION

The multiplication of two operands with following addition of third operand (Fused Multiply-Add, FMA) had become the standard operation of the modern processors. Often, however, a difference between the product of first two operands and third operand is also calculated together with the sum.

Many implementations of the synchronous FMA units are known in a technical literature [1, 2, 3]. The asynchronous FMA unit implementations are less explored [4, 5]. And almost no implementations of self-timed devices of this type are known, which would match the features of the Speed-Independent (SI) circuits whose correct functioning does not depend on the delays of the circuit's elements.

SI circuits reduce energy consumption due to non-use of clock generator and "clock tree", as well as due to automatic switching hardware part not used in the current cycle of information processing into an energy-saving mode. In addition, the SI circuits keep their correct workability at ultra-low supply voltages. This opens up the broad prospects for designing energy-efficient devices. The tradeoff for such benefits is hardware redundancy and overhead delays because of an additional indication subcircuit and space phase in a work cycle of the SI. However, in some circuit classes, such as fault-tolerant ones, this redundancy turns out to be insignificant, and in some cases [6] SI circuits have even less complexity than their synchronous analogs.

* The reported study was partially funded by Presidium RAS Program №27 (project 0063-2018-0003) in IPI FRC CSC RAS.

This paper outlines the approaches to designing 64-bit unit of a gigaflops range that belongs to SI circuits, performs FMA operation, and complies with IEEE 754 Standard, taking into account its preferred characteristics.

II. INCREASING PERFORMANCE OF THE SIFMA

Until recently, hardware developers directed their efforts to achieving maximum performance. While low bit-width SI circuits provide increased performance in 1.5-1.7 times compared with synchronous analogs [6], high bit-width SI classic circuits are often worse than their synchronous analogs for this parameter. The vast majority of SI units utilizes a two-phase switch discipline. The operation execution cycle includes the work phase, during which the SI circuit calculates the result of the operation, and spacer phase, during which the circuit is getting ready for next operation. As a result, a time elapsed by a particular operation increase not less than by two times. In part, this loss time in comparison with synchronous circuits is leveled due to excessive duration of a clock period in the synchronous circuits targeting the worst case operating conditions.

A typical solution of the task of accelerating any digital device is pipelining successive stages of data processing. FMA implementation variant with maximum speed was developed and presented in [7]. Fig. 1 illustrates its flowchart. It contains two FMA cores working in parallel. The maximum speed was achieved due to doubling hardware. FMA was designed in 65 nm CMOS process. It contains 4 pipeline stages, as Fig. 2 demonstrates, and provides performance up to 3 Gflops.

An indication of all transient processes termination in all circuit elements is a very serious problem for SIFMA. This should be performed twice in one cycle of the SI circuit to indicate the completion of the work and spacer phases. A complexity and a delay of an indicator subcircuit depend on complexity of the indicated circuit.

The most complicated and latent functional block in the SIFMA is a hardware multiplier of two operands, summarizing the partial products with high bit width. This paper studies the different implementations of the single-channel FMA version providing maximum performance or minimum complexity.

 978-1-7281-0339-6/19/$31.00

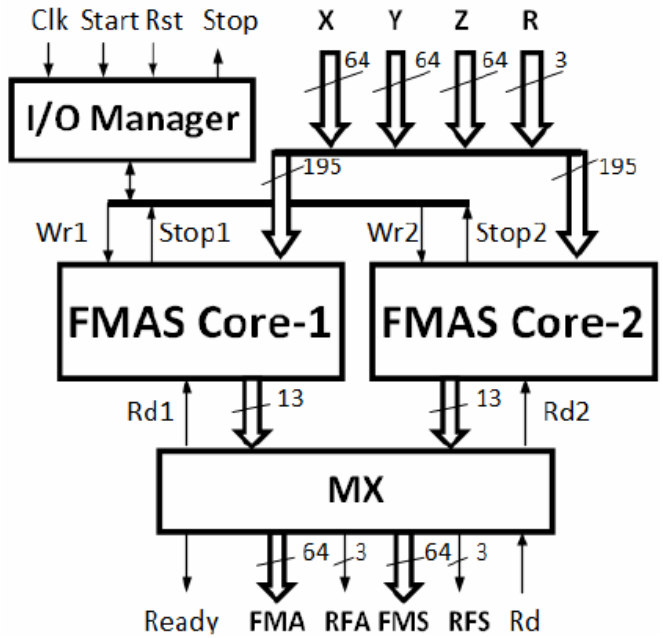

Fig. 1. Flowchart of SIFMA with maximum performance

Table I shows the simulation results for SI multiplier of the mantises of 64-bit operands using the modified Booth algorithm with dual-rail signals and Wallace tree based on adders with ternary self-timed signal encoding.

Simulation was performed in ModelSim for 65 nm CMOS process based on standard cell library for typical conditions: supply voltage of 1V and the temperature of 25 Celsius degrees. A tree of three-input hysteresis triggers (H-triggers) provides an indication of the entire multiplier. Presented data correspond to a work phase. Duration of a spacer phase is higher approximately by 30%. The overall cycle time is about 1.6 ns and approximately 2 ns taking into account the layout implementation.

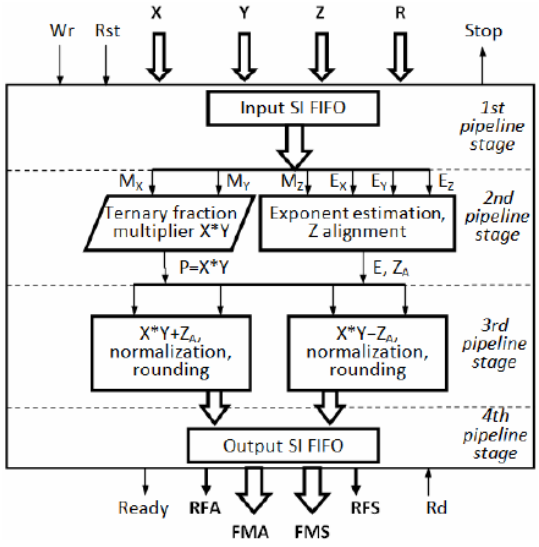

Fig. 2. Flowchart of the SIFMA core

TABLE I. WORK PHASE TIMINGS OF SIFMA MULTIPLIER

Functional component	Number of Outputs	Work phase duration in one pipeline stage[a], ps	Indication timings[a], ps
Booth decoder	81	40	200
Booth encoder	1512	70	300
Dual-rail to ternary code converter	756	100	300
1-st stage adders of the Wallace tree			
- layer 1	406	120	330
- layer 2	406	140	350
- layer 3	406	170	380
2-nd stage adders of the Wallace tree			
- layer 1	198	190	400
- layer 2	198	210	420
- layer 3	198	230	440
3-rd stage adders of the Wallace tree			
- layer 1	164	260	470
- layer 2	164	280	500
- layer 3	164	300	530
4-th stage adders of the Wallace tree			
- layer 1	108	320	530
- layer 2	108	340	550
- layer 3	108	360	580
Output register	108	400	610

[a]. All timing intervals start at acquiring corresponding input data

Table I demonstrates that the direct computations in the full operating cycle of the pipeline stage require approximately 900 ps without taking into account a layout, or about 1.2 - 1.3 ns with parasitic components extracted from the layout.

It is obvious, that for maximum performance, one should split the multiplier at least onto two pipeline stages. Choosing a manner of a hardware separation is a compromise between performance and redundancy of the multiplier implementation. And a necessity of an indication of all initiated transients termination for all cells in each pipeline stage affects this choice significantly.

III. PIPELINING MULTIPLIER

Intermediate data registers complete each pipeline stage. They have a bit width corresponding to the number of its output variables multiplied by the code length presented these variables. Table I illustrates a hardware cost of their implementation, and the times of indication completion reflect the duration of the individual parts of the pipeline stage. Significant hardware and time resources are required for the Booth coder implementation. Indication of the converter of the dual-rail self-timed code into ternary representation for a difference between two dual-rail signals ends simultaneously with the indication of the Booth coder work ending. Therefore, the minimum duration of the work phases of first pipeline part without taking into account a time for writing data into the output register is about 300 ps or 360 ps with a glance to the layout.

Obviously, this is a minimal set of functions for one pipeline stage, and it corresponds to maximum performance of the SIFMA multiplier. The remaining Wallace tree operations can be implemented as one pipeline stage or split into two stages for maximum performance. In the second half of the Wallace tree, the number of elements is being significantly

reduced, so that a time for their indication becomes uncritical for working-time of the pipeline stage.

TABLE II. THREE-STAGE PIPELINE TIMINGS OF SIFMA MULTIPLIER

Functional component	Work phase, ps		Spacer phase, ps		Total stage duration, ps
	Logic part	*Indication part*	*Logic part*	*Indication part*	
Booth decoder and encoder, dual-rail to ternary converter	140	370	180	480	850
First two Wallace tree layers	170	370	220	480	850
Last two Wallace tree layers	170	380	220	490	860
Entire multiplier unit	480	1120	620	1450	2560

The version with peak performance requires three pipeline stages for the multiplier implementation. Table II presents the main characteristics of the corresponding SIFMA multiplier (based on simulation considering output register, but without taking into account the layout). They allow for concluding that only in very neat layout implementation, the maximum performance of the 64-bit single-channel SIFMA can reach 3 Gflops in 65-nm CMOS process. This is achieved by increasing latency (the three-stage pipeline) and by using additional registers for data storage (about 3000 SI triggers).

If additional latency and hardware costs are not desirable, one should prefer one-stage multiplier implementation calculating product for increased by 20-30% time interval. The following techniques improve performance and reduce latency of SIFMA multiplier:

- Forced accelerated switching Wallace tree adders into spacer phase.
- Providing simultaneous switching Booth encoder together with some part of the Wallace tree layers into spacer phase and executing complete work cycle by remaining Wallace tree layers.

The forced acceleration of the spacer phase reduces its duration when two conditions are met:

- Presence of some sequence layers switch in this pipeline stage when elements when forced switching to spacer is carried out on a number of consecutive elements simultaneously.
- Correct layout implementation of such forcing net because of a large number of simultaneously affected logical elements.

Accommodation of the multiplier in one pipeline stage just implies long-term sequence of elementary logic operations during calculations requiring indication. This allows one to reduce indication timings essentially. Indication of a significant part of first Wallace tree layers is completed together with the end of the computing process, and one should indicate only elements from last Wallace tree layers and an output register completing their work.

Besides, a long switch sequence allows one for taking advantage of the SI implementation feature, namely a single spacer state of all its elements in the current stage, and for organizing their accelerated switching into this state through a few specially selected locations of this sequence. This is available due to a common indication of all cells at the end of the spacer phase and allows for reducing stage delay by 30-40%. Dual-rail to ternary self-timed code converter outputs together with first three Wallace tree layers outputs were chosen as such enforcement points as Fig. 3 shows. Here A and B are multiplicands, Ack_1 and Ack_2 – acknowledge signals, Rq_1 and Rq_2 – request signals, FTS_1 and FTS_2 – signals that enforce switching both halves of the SIFMA multiplier into spacer phase. Acknowledge and request signals provide two-phase discipline for SIFMA.

To speed up the work phase in this case, an additional register on base of H-triggers is used at the output of second Wallace tree layer. This register records the successful completion of the Booth encoder and first two Wallace tree layers, and allows them to begin switching into spacer phase until the work phases in the multiplier will complete. At the same time, the work phase of third and fourth Wallace tree layers is initiated followed by their spacer phase. Output stage register records the work phase result for entire multiplier and lets intermediate register to switch into spacer phase at the output of second Wallace tree layer. This method is similar to implementation of an internal local pipeline with a special discipline inside the main pipeline stage. It reduces an overall multiplier work cycle time by another 20-25% due to increasing multiplier complexity by approximately 10%.

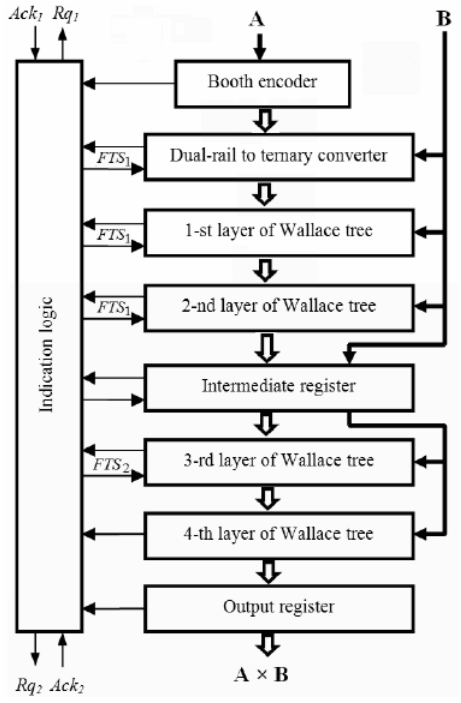

Fig. 3. Forced transition to spacer in SIFMA multiplier

TABLE III. SIFMA MULTIPLIER IMPLEMENTATIONS

Variant	Pipeline stages	Latency, ns	Complexity (CMOS transistors)
Maximum performance	3	1.0	410000
Minimal complexity	1	2.6	190000
Combined unit	1	1.3	320000
Synchronous base unit	1	1.3	125300

An additional reduction of hardware complexity in this case is achieved by partitioning the input data array onto least significant and most significant input arrays and consistent processing them through entire set of computational procedures on the same hardware resources with the half bit width.

Table III demonstrates the research results of the SIFMA characteristics and their comparison with the basic synchronous variant [3], designed in the same CMOS process. Latency estimates were obtained by means of the functional simulation of the designed variants with Modelsim program on base of a typical standard cell library for 65-nm CMOS process. The library was supplemented by a number of the developed and characterized SI cells.

Analysis of Table III proves that depending on the main target criteria of the SIFMA unit one can use either multiplier with a large latency, but with minimal complexity and, accordingly, with minimum energy consumption, or multiplier with lowest latency, but with a maximum complexity and great power consumption.

IV. INPUT AND OUTPUT FIFO IMPLEMENTATION

The FIFIO principle in electronic circuits is typically used to buffer data flow between communicating units to synchronize an interchange speed with data processing speed. FIFO is often implemented as a virtual ring queue (VRQ) of the dual-port SRAM, where one port is used for writing and another is used for reading. For large volume FIFOs, VRQ is the only possible and the most energy-efficient: writing into FIFO initiates transients only in one cell corresponding to the write counter pointer. There are three types of FIFO organization as VRQ:

- Synchronous FIFO using a single clock for both reading and writing.
- Asynchronous FIFO using different clocks for reading and writing. This often leads to metastability.
- Self-timed FIFO using request and acknowledge signals between interacting components on basis of indicating completion of their transition processes instead of usage any clock.

Speed-independent (SI) FIFO implementation assumes solving the following problems:

- Choosing and implementing the type of SI encoding both for write and read operations to/from a specific FIFO cell.
- Addressing specific FIFO cells for these procedures.

Traditional SI solution of these problems come to using dual-rail with a spacer encoding for input, intermediate and output FIFO interfaces that doubles the number of communication lines and increases the hardware complexity both of a separate cell, and of a mechanism for its access.

A small volume of the SIFMA FIFO (no more than four) makes it reasonable to implement its one bit as a shift register (SR) with input and output heads. In doing so, the access mechanism becomes unnecessary, and hardware resources for its indication also are reduced dramatically. The previous version of FIFO within the SIFMA described in [7] was implemented on the basis of semi-dense register [8], which involves alternating SR stage storing the useful information and service stages intended for the synchronization of the passing information from the input head to the output head. Fig. 4 shows a flowchart of the semi-dense SI register, when input data ("Data") are provided by a synchronous source. Here "Clk" is a clock signal from the synchronous source; "Ready" reflects the data readiness at the FIFO output head; "Read" acknowledges FIFO that data were read from it; "IM Cell" is intermediate FIFO cell.

Simplicity of the semi-dense register implementation and behavior is accompanied by a substantial increase in the number of its intermediate signals and rather a large complexity, because only each second stage of FIFO stores real data as FIFO is "semi-dense".

So a new FIFO implementation was developed and used in SIFMA. It is founded on a SR file with parallel writing and reading. Fig. 5 shows its structure. Input data come to an input head ("IH") that distributes them to the next stage of SR, and W/R_i signals from "Control Unit" allow for writing data "DIn" into regular SR stage. Control Unit detects a state of each stage or the SR and allows for shifting data from the input head of the SR to its output head. It also forms "Stop" signal that notifies data source about fullness of the FIFO, and advises data receiver of data "DOut" readiness. The Control Unit is common for all bits of the FIFO. So its complexity slightly affects total complexity of the FIFO.

Table IV shows the pin amount and complexity for both variants of one FIFO bit designed for storing 4 operands. The proposed new SR version has by 2.3 times less pins than the semi-dense register. Moreover, proposed variant has by 1.8 times lower complexity.

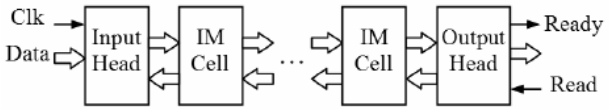

Fig. 4. Semi-dense FIFO

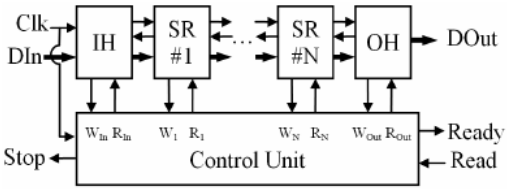

Fig. 5. SI FIFO on base of shift register

TABLE IV. FEATURES OF TWO FIFO RELEASES

Version	Pin number	Complexity (CMOS transistors)
Semi-dense FIFO (release #1)		
Input head	11	56
Intermediate cell	13×5	32×5
Output head	12	42
Total	88	258
Proposed FIFO (release #2)		
Input head	5	38
Intermediate cell	11×2	34×2
Output head	11	34
Total	38	140

The input head of the release #1 assumes the usage of classic SI exchange protocol between the FIFO and data source. After writing data into FIFO and forming indication signal reflecting a successful end of the work phase, an input must remain unchanged until an opposite switching indicator signal caused by transition of the write permission signal. However, if the source of the information does not meet the requirements of the SI protocol (for example, implements a synchronous communication protocol), FIFO information may be rewritten by an incorrect data. The input head of the release #2 provides SI work mode within both the synchronous, and SI protocol.

Thus, the release #2 of FIFO implementation ensures its reliable operation with significantly lower hardware cost than release #1. This contributes to improve its energy efficiency.

V. SIFMA'S ENERGY EFFICIENCY

Over dual-phase work discipline and redundant coding all data signals, an arbitrary half of combinational elements in the SI circuits switch twice during work cycle. Due to this, combinational SI circuits demonstrate higher power consumption versus their synchronous analogs at the same performance.

However in practice, the average active work time of the FMA in general purpose multi-core CPUs does not exceed 5-10% and only in rare cases can reach 30%. So in synchronous implementations of the FMA unit, its clock tree causes odd energy consumption during inactivity time interval. To prevent this, clock tree can be turned out for the most part of work time as it is not needed. There are some techniques in synchronous circuitry, e.g. "clock gate", providing adaptive control of the clock tree activity individually for each functional unit in a computational system. But they require the additional hardware resources and lead to some delay in data processing pipeline needed for turning clock subcircuit on/off.

On the contrary, SI circuits utilize real work activity of their functional blocks naturally due to two-phase interaction protocol between these functional blocks. This feature makes any SI circuit, and particularly SIFMA, preferable for usage in hardware with low computational activity.

Among considered SIFMA multiplier implementations, the one-stage multiplier is the most energy-efficient because of its minimal hardware cost. Economical SIFMA unit based on one-stage SI multiplier implementation is comparable to the basic standard synchronous solution at performance and hardware costs and has significantly less power consumption due to the lack of losses because of the clock tree. Besides, SIFMA provides any constant fault detection due to fixing it by the indicators directly during the operating.

VI. CONCLUSIONS

Development of the SI circuits is always a compromise between their performance and complexity. The necessity of an indication of all cells in the designed SI circuit significantly affects this compromise.

The peak performance of the 64-bit SIFMA in 65-nm CMOS process can reach 3 Gflops in its single-channel version subject to three-stage pipeline implementation of the SI multiplier, while the single-stage multiplier release provides the best "performance/complexity" ratio and the highest energy-efficiency.

Proposed few-stage SI FIFO on basis of SI shift register provides the best characteristics and higher energy-efficiency in comparison with classic semi-dense shift register. Multistage SI FIFO requires an additional study.

REFERENCES

[1] R.V.K. Pillai, S.Y.A. Shah, A.J. Al-Khalili, and D. Al-Khalili, "Low power floating point MAFs – A comparative study", Sixth International Symposium on Signal Processing and its Applications, Kuala Lumpur, 2001, V. 1, pp. 284-287.

[2] 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.

[3] E. C. Quinnell, "Floating-Point Fused Multiply-Add Architectures", PhD Thesis, The University of Texas at Austin, May 2007. – 150 P. URL: https://repositories.lib.utexas.edu/bitstream/handle/2152/3082/quinnelle60861.pdf (Retrieved 2018-12-11)

[4] J.R. Noche, and J.C. Araneta, "An asynchronous IEEE floating-point arithmetic unit", Science Diliman, Philippines, 2007, V.19, No.2, pp. 12–22.

[5] R. Manohar, and B.R. Sheikh, Operand-optimized asynchronous floating-point units and method of use therefor, US patent, № 20130124592. May 2013.

[6] Yu.A. Stepchenkov, Yu.G. Diachenko, and V.S. Petrukhin. "Experience in developing self-timed microcontroller core on a base of gate array" // Nano- and Microsystems technique, №5, 2006., p.29-36 (in Russian).

[7] Yu.A. Stepchenkov, V.N. Zakharov, Yu.V. Rogdestvenski, Yu.G. Diachenko, N.V. Morozov, and D.Y. Stepchenkov, "Speed-Independent Fused Multiply Add and Subtract Unit" // Proceedings of IEEE EastWest Design & Test Symposium (EWDTS'2016), Yerevan, October, 14 - 17, 2016. P. 150-153.

[8] V.I. Varshavskij et al., "Automata control of concurrent processes in computers and discrete systems". Moscow, "Nauka", 1976, 400p. (in Russian).