Quasi-Delay-Insensitive Computing Device: Methodological Aspects and Practical Implementation

Yuri Stepchenkov, Yuri Diachenko, Victor Zakharov, Yuri Rogdestvenski, Nikolai Morozov, Dmitri Stepchenkov

Institute of Informatics Problems, Russian Academy of Sciences, Vavilova 44, build 2, Moscow, 119333 Russia
{YStepchenkov, YDiachenko, VZakharov, YRogdest, NMorozov, DStepchenkov}@ipiran.ru

Abstract. The approaches to self-timed hardware design are presented. The conditions of intersystem integration of synchronous and self-timed devices are considered through the example of the quasi-delay-insensitive computing device development. This device performs functions of division and square root extraction. It operates with numbers of single and double precisions corresponding to the IEEE 754 standard.

Keywords: self-timed, quasi-delay-insensitive, division, square root, Radix-2

1. Introduction

Synchronous, asynchronous and quasi-delay-insensitive (QDI) methodologies are the alternative approaches to the digital hardware design [1]-[3]. Each of them has its advantages and shortages which determine their appropriate applications. At present QDI-methodology is used not so widely as traditional synchronous and asynchronous ones due to some reasons, such as relatively higher labor-intensiveness for the design of QDI-circuits and some hardware superfluity.

But it also provides such features as enlargement of the capacity for work area in the range of changing voltage and environment temperature and also stable operation despite any finite delays in elements. This allows them to operate when voltage and environmental conditions are close to the boundary, determined by physical properties of semiconductor active elements [4, 5] – transistors, if this is not critical in respect to the device performance. QDI-circuits are correctly operating with the *maximal* possible for current conditions performance. At the same time for a correct operation of synchronized devices, the period of clock signal is chosen taking into account *the worst case* – maximal possible time for the elements switching under unfavorable conditions (supply voltage, temperature, etc.). Thus the price of the correct operation for synchronous devices is the underexploitation of its abilities regarding performance, compared with a realistically possible one provided by QDI-methodology.

The increase in the hardware outlays (number of transistors in the circuit) is especially evident during the implementation of combinative units, but it is graded by the absence of a synchronization circuit and in some practical cases proved to be

insignificant (for example, during implementation of sequential devices) [4]. Besides the obtained enlargement of the capacity for work area (at first in voltage) compensates the circuit complication in devices designated for work with a restricted power supply.

Modern problems of VLSI-systems design can be more efficiently solved by moving to the synchronous-self-timed style. This style combines the advantages of all three methodologies of digital hardware design.

This paper presents the results of the development of a QDI-device which performs functions of division and square root extraction. It operates with numbers of single and double precision corresponding to the IEEE 754 standard [6]. The calculations are implemented with respect to the standard but with some simplifications:

- only normalized operands arrive to the input of the computing device;
- the result is also a normalized number;
- in case the result cannot be represented as a normalized number, an exceptional situation takes place.

According to the IEEE 754 standard operands are represented as floating point numbers and consist of sign bit, exponent area and mantissa area. The main complexity consists in calculation of the result's mantissa. Later, with the calculation result, we will average its mantissa calculation.

At present, many algorithms for calculating the results of division and square root extractions are known [7]-[13]. But not all of them are suitable for efficient usage of QDI-schematics. The proposed computing device provides optimal balance of performance and hardware expenditures with saving all advantages of QDI-circuits.

2. Methodological Aspects

The feature of QDI-circuits – request-acknowledge interaction between blocks, adjacent in the line of information processing – makes multi-stage implementation of any computational algorithm the most efficient, when the control is transferred from the previous stage to the next one asynchronously, according to the real performance of each stage. The most efficient advantages of QDI-devices are that they show their worth in case of pipeline implementation, and the number of stages in the pipeline should be no less than three. This is determined by the order of QDI-devices operation [3]:

- the presence of two phases in each QDI-device: working (active) and spacer (pause);
- the usage of request-acknowledge interactions between adjacent devices in the data processing pipeline.

The first condition is necessary for the successful implementation of absolute control on the termination of transition processes in each device – stage of pipeline. The second condition provides the strict sequence of switches for the adjacent pipeline stages, guarantees trustworthiness of data for the input of each stage at any moment and QDI-circuit workability with any finite delays of the elements included in the circuit.

According to the principles of self-timing, a QDI-device cannot start switching to the next phase until the previous device reaches a similar phase, and until the next device reaches the opposite phase of operation. The more stages in the pipeline, the less total unproductive time spent when each device in the joint system is waiting for

permission from the adjacent devices to move to the next phase of work. Practically such interaction is implemented by means of hysteretic triggers [3] or C-elements, inputs for them are indication outputs of the previous and next devices, which prove the termination of a corresponding device transition into the next phase of operation.

Taking into account features of pipeline stages operation and QDI-basis for their implementation it is possible to use input control signals as inputs of indicators. Fig.1 illustrates the scheme of interaction of three adjacent stages in QDI-pipeline of the described computing device. The connecting element is C-element. Enable input (*E*), providing change of operating phases of QDI-device in the i-th stage, is forming by C-element basing on indicative output (*I*) of the next (i+1)-st stage and control signal from the previous stage. Indicative output *I* proves the termination of the stage switch to the corresponding operating phase. As the result the strict sequence of adjacent stages switches is provided. In this point the proposed solution differs from similar implementations [12]-[13], where the control signal of the i-th stage (*Reset*) is formed only basing on information about termination of switch to the current operation phase from the next stage. In case of some definite delays of the circuit elements in the different stages such discipline can lead to the wrong operation of the device, for example in the following hypothetic situation: i-th stage terminated switch to the working phase, formed data signals for the (i+1)-st stage and signal $Reset_i$ for (i–1)-st stage, which quickly switches to the spacer phase and then to the working phase, renewing data signals at the inputs of i-th stage. If to that time (i+1)-st stage will not be able to form control signal $Reset_{i+1}$, for the i-th stage, then the data at the output of the i-th stage will be changed, and that can result in the calculation error on (i+1)-st stage. The proposed implementation is free from such shortcoming.

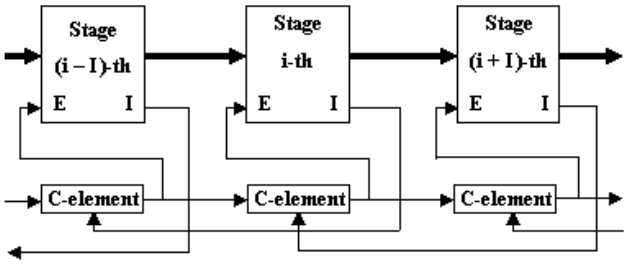

Fig. 1. Arrangement of request-acknowledge interaction between stages of computing device

The number of stages in the pipeline implementation of the algorithm can be arbitrary but it is restricted by the requirements to the overall size of topological implementation and power consumption. Theoretical analysis shows [12] that the optimal (in respect to performance) number of stages in such an algorithm are equal to 4 or 5. In such case, the pipeline becomes balanced in the best way.

In QDI-circuits it is necessary to indicate the termination of each element switch to catch timely the appearance of a fault in the circuit and stop calculations. But practical implementation of the full-scale indication leads to the big overheads – up to 50% of the circuit own complexity. That is why in practice it is usually used the principle “of the expected delays relation for the elements fabricated in one chip”.

Such approach allows to reduce essentially the expenditures on the circuit indication and accelerate its operation. In the most practical cases not expected on the operation in extreme conditions such approach is sufficient to provide fault-tolerant circuit operation in the wide range of work conditions. And only one requirement remains unchanged – implementation of request-acknowledge interaction between adjacent blocks.

The most fast-acting algorithms of division and square root extraction are based on the usage of multiplication (for example, the Newton-Raphson method [8]) or on table methods [7]. But they require implementation of a multiplier in hardware or rather big volumes of ROM, which in the implementation of QDI-circuitry leads to considerable hardware expenditures.

This is why we chose an irretrievable algorithm of division SRT Radix-2 [12, 13] as the basic one. It allows combining functions of division and square root extraction in one unit with minimal additional expenditures. It is important in the implementation of digital devices with QDI-basis.

The property of self-timing is brought to the algorithm by the paraphase discipline of all data signals coding, indicating moments of termination of all transition processes and the arrangement of the request-acknowledge interaction between adjacent blocks in a computing device. Similar facilities have been just used in projects [12, 13]. But they did not meet the requirements of QDI-methodology; when used, their circuits of request-acknowledge interaction are not figured on the possibility of wide dispersion in the delays of the same type elements due to the degradation of active structure parameters or due to the local deviation of technological parameters.

3. Flow chart of the computing device

Based on the given overall dimension restrictions and requirements to the performance of the computing device we chose the four-stage pipelined variant of its implementation. A flow chart of each stage is presented in Fig. 2.

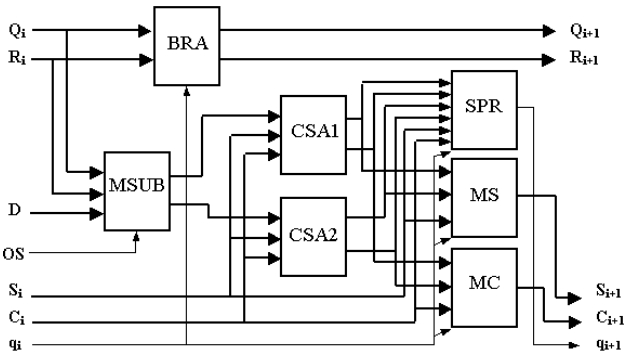

Fig. 2. Flow chart of one stage

The stage contains the following main blocks:

- block of the result accumulation (BRA);

- multiplexer of subtrahend in recurrent formula (MSUB);
- Carry-Saved Adders (CSA1, CSA2);
- selector of partial result (SPR);
- multiplexer of sum (MS);
- multiplexer of carrying (MC).

General flow chart of computing device is represented in Fig. 3. Besides four similar stages QDI1 – QDI4 it contains also blocks:

- input register of operands and features of operation (IR);
- input multiplexer of operands (MO);
- indication circuit (IC) with control circuit (CC);
- circuit of exponents processing (CEP);
- block of mantissa post processing (PP);
- output register of the result (OR).

Dimensions of operands in Fig. 3 correspond to the asynchronous variant of a computing device implementation. For QDI-variant the dimensions of the external signals remain unchanged as this computing device is designated for work with synchronous environment. The dimensions of the internal buses are increased twice due to the paraphase presentation of the data signals [3]. That allows encoding two working and one space states for each data signal. In this case spacer can be zero (00) or one (11). In our implementation of computing device only zero spacer is used for all internal paraphase signals.

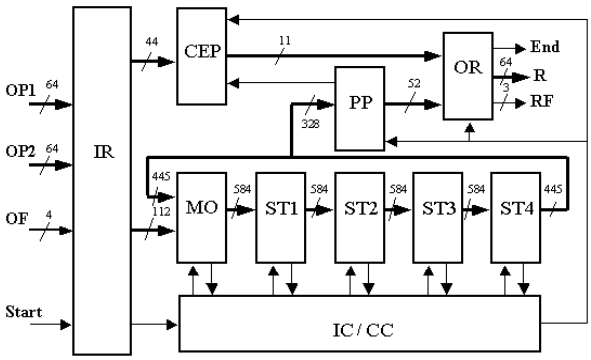

Fig.3. Flow chart of the computing device

Blocks IR and OR are implementing an asynchronous interface with a synchronous environment. The input signal *Start* is used for the synchronization of writing to the register IR of input data: operands *OP1* and *OP2* and features of operation *OF*, after which the circuit starts to operate without any assistance. The result of the operation together with flags of exceptions are written to OR register. After termination of forming the result and writing it to the output register OR the ready flag *End* is raised. On this sign, the synchronous environment of the device detects the moment when the result and flags of the exclusive situations can be read from the OR register.

The indication circuit IC covers by its signals all blocks coordinating together with CC their operation.

4. Practical implementation of computing device

One of the problems of multi-stage calculations is the saving of intermediate result. In synchronous implementations it is traditionally solved using registers separating adjacent stages. In QDI-circuits the problem of intermediate result saving cad be solved by using elements with memory. In such case it is efficient to use for the computing device implementation the elements with dynamical logic [12]. Paraphase discipline of data signals simplifies the transition of such elements to the spacer and protects from the appearance on the output false short-time working states while switching to the working phase.

To implement the divider we used the library of standard elements of Artisan Company [14]. It was necessary to additionally design 27 pseudo-dynamic elements, which provide saving of intermediate data, forming of paraphase data signals and simplify indication of the transition processes termination while switching stage to the next operating phase. They have been implemented topologically in the style of standard library elements.

The example of pseudo-dynamic element implementing function of the multiplexor 2:1 is presented in Fig. 4. It has paraphase data inputs {*A, AB*}, {*B, BB*}, inphase select signals *EA, EB*, inphase control signal *E* and paraphase output {*Y, YB*}. A transition to spacer is provided by arriving control signal *E*=0. In such case the outputs are switched to the state *Y*=*YB*=0. By using weak transistors WT0 and WT1 outputs can support spacer state for without limit long time after transition of control signal to the state *E*=1, until working combination will arrive to the data inputs and select inputs, which switch one of the multiplexer's outputs to the state of logical 1.

Saving of the working state at the outputs is not supported but it corresponds to the zero potential at the inputs of invertors forming multiplexer's outputs and due to the parasitic capacitances can be saved for a rather long time. Besides the scheme operation is arranged in such way that input *E* switches to the working state shortly before forming working combination at another inputs.

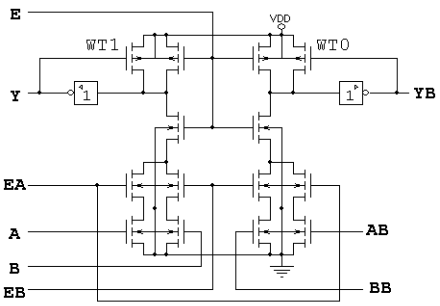

Fig. 4. Schematic diagram of pseudo-dynamic multiplexer 2:1

Similarly the circuits of other pseudo-dynamic elements are constructed. The usage of control signal *E* essentially accelerates transition of such element to the spacer state. But in case of multiple-order architecture of computing device they are strongly loaded and require powerful drivers. That is why in some library elements, which do not require the alignment of the execution time of similar elements in multiple-order

architecture of a computing device, the pseudo-dynamic elements have been used, in which the transition to spacer is implemented by the combination of some input data signals. The example of such element is XOR3M (see Fig. 5).

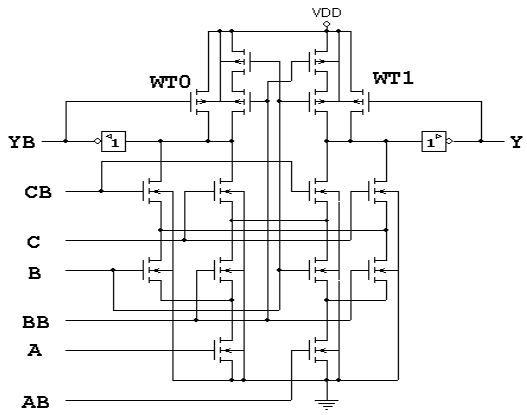

Fig. 5. Schematic diagram of pseudo-dynamic element XOR3M

QDI computing device have been implemented within test chip with standard 0.18μm CMOS-technology with 6 metal layers [14]. According to the required specification, the area of the computing device should not exceed 0.36 mm^2. In the result of thorough manual topological design in CAD CADENCE, the block of the divider is formed in a rectangle with an area of 0.33 mm^2 and a ratio of sides 1.4:1 (Fig. 6).

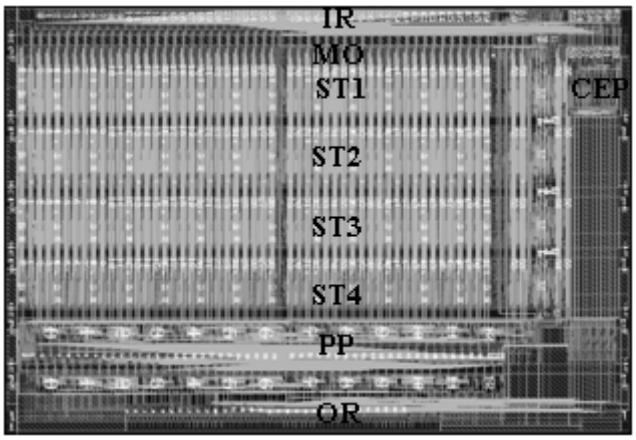

Fig. 6. Topological implementation of computing device

In the Table we represent the averaged data at the time of execution for the operations of division and square root extraction for simulating limited random set of operands and rounding modes. Nevertheless, all complicated cases of the rounding according to the requirements of the IEEE 754 standard, found themselves in this set.

Table. Result of simulation with Ultrasim

№	Conditions of simulating , Udd, T^O C	Division, ns		Square root, ns	
		Precision of calculation			
		single	double	single	double
1	1,98 V, - 60^0 C, best	20.6	34.7	22.0	36.9
2	1.8 V, + 25^0 C, typical	28.7	46.7	30.4	49.1
3	1.62 V, + 125^0 C, worst	38.9	63.9	40.0	70.3
4	0.9 V, + 125^0 C	139	219	125	199
5	0.8 V, + 125^0 C	185	300	172	276
6	0.7 V, + 125^0 C	290	480	265	422
7	0.6 V, + 125^0 C	536	858	491	775
8	0.5 V, + 125^0 C	1293	2100	1209	1893
9	0.4 V, + 125^0 C	4656	7682	4325	6940
10	0.35 V, + 125^0 C	12920	21705	9142	14440

The data in the Table confirms that the times of execution by the computing devices of both operations practically coincide, and the typical corner and computation with double precision are about 50 ns. It can be seen from the Table that the performance of the computing device within the required specification varies in a wide range: the best and the worst execution times differ almost by two times.

The computing device, as each QDI-device, is characterized by the steady capacity for work in case of reduced voltage, for example, while falling battery voltage lower than acceptable norms (see lines 4 – 10 of the table).

For those applications where the decisive factor is the keeping capacity for work even owing to the essential falling of the performance, usage QDI-devices becomes of current importance. The results of the computing device measurements within test chip confirmed data obtained while simulating.

Within the test chip, three 64-bit homogeneous devices have been implemented, the described multi-stage QDI-variant and two synchronous variants. The comparative results of all three variants of computing device testing are presented in Fig. 7.

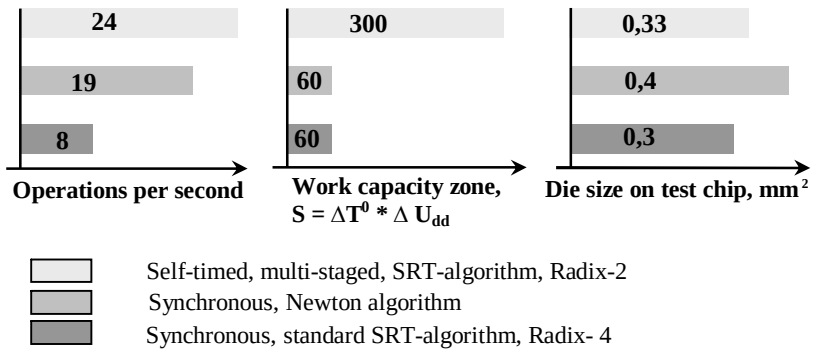

Fig.7. Results of different variants of computing device implementation testing

Presented results regarding performance are obtained for the following conditions:

- the probability of the execution of division and square root extraction operations with single and double precision are the same;
- the number of operations executed in the “best” conditions – 25%, in “typical” conditions – 50%, in “worst” conditions – 25%;
- the set of operands is statistically significant.

At present we are developing the completely QDI-variant of a computing device. In it we obey all principles of QDI-circuits design, which provide the real independence on the elements delay in any conditions of device exploitation and all advantages of QDI-circuits in full volume.

5. Summary

The development of QDI computing device for division and square root extraction is demonstrated:

- using the methodology of QDI-circuit design allows obtaining viable and efficient solutions even for a mainly combinative device such as a divider, though traditionally combinational circuits are the least “convenient” for implementation with an QDI-basis;
- the efficient solution of computing devices with QDI-circuits basis is possible only on the base of multistage – pipeline – organization of a computational path; in our case – using four stages of the same type, each of them calculates one bit of the result;
- proposed implementation provides the same performance while execution of both operations – division and square root extraction;
- complex comparative parameters: for performance and zone of capacity for work and area occupied on the chip, the presented QDI-variant of the computing device exceeds synchronous analogues.

References

1. Varshavsky V., “Time, Timing and Clock in Massively Parallel Computing Systems", *Proceedings of International Conference Massively Parallel Computing Systems*, April 6-9 1998, Colorado Springs, USA, Special Session 2.
2. P. Beerel, J. Cortadella, and A. Kondratyev, “Bridging the gap between asynchronous design and designers (Tutorial),” in *VLSI Design Conference*, (Mumbai), 2004.
3. Varshavsky, V.; Kishinevsky, M.; Marakhovsky, V. at al., *Automata Control of Concurrent Processes in Computers and Discrete Systems*, Ed. by Varshavsky, V., Moscow, Nauka, 398 pp., 1986 (in Russian) (English Translation - Self-Timed Control of Concurrent Processes, Kluwer Academic Publishes Groop, 428 pp., 1990).
4. Sokolov I.A., Stepchenkov Yu.A., Petrukhin V.S., Djachenko Yu.G., Zakharov V.N. Self-Timed Circuitry – Perspective Method of Hardware Development // High Availability Systems, v. 3, № 1-2, 2007, p. 61-72.
5. L. Plechanov, Yu. Stepchenkov. Experimental test of some features of strictly self-synchronized electronic circuits. //Annual “The Systems and Means of Informatics”, Issue 16, 2006, Moscow, Nauka. – P. 445-452. (In Russian).

6. IEEE Standard for Binary Floating-Point Arithmetic./IEEE Std. 754. New York ANSI—1985, Aug.
7. S. E. McQuillan and J. V. McCanny, “Fast VLSI algorithms for division and square root”, J. VLSI Signal Processing, V. 8, - P. 151–168, Oct. 1994.
8. P. Montuschi, L. Ciminiera, and A. Guistina, “Division unit with Newton-Raphson approximation and digit-by-digit refinement of the quotient,” IEEE Proceedings: Computers and Digital Techniques. – 1994. - V. 141. - P. 317-324.
9. Lang T. and Montuschi P. Very-high radix combined division and square root with prescaling and selection by rounding / In Proceedings of the 12th IEEE Symposium on Computer Arithmetic. - 1995, July. - P. 124–131.
10. J. Arjun Prabhu and Gregory B. Zyner. 167 MHz Radix-8 Divide and Square Root Using Overlapped Radix-2 Stages / In Proceedings of the 12th Symposium on Computer Arithmetic. – 1995. - P. 155-162.
11. Ajay N., Atul D., Warren J., Debjit D. S. 1-GHz HAL SPARC64 Dual Floating Point Unit with RAS Features / In Proceedings of the 15th IEEE Symposium on Computer Arithmetic. - 2001.
12. T.E.Williams. M.A.Horowitz. A Zero-Overhead Self-Timed 160-ns 54-b CMOS Divider / IEEE Journal of Solid-State Circuits. - 1991. - V. 26. - №11. - P. 1651-1661.
13. G. Matsubara, N. Ide, H. Tago, S. Suzuki and N. Goto. 30-m 55-b Shared Radix 2 Division and Square Root Using a Self-Timed Circuit / In Proceedings of the 12th Symposium on Computer Arithmetic. - 1995. - P. 98-105.
14. Chartered Semiconductor 0.18µm IB Process 1.8-Volt SAGE-XTM. Standard Cell Library Databook / Artisan Components, February 2003, Release 1.0.