DE GRUYTER OPEN

Journal of ELECTRICAL ENGINEERING, VOL. 66, NO. 6, 2015, 329–333

MODULAR DESIGN OF FAST LEADING ZEROS COUNTING CIRCUIT

Nebojša Z. Milenković — Vladimir V. Stanković — Miljana Lj. Milić [*]

In modern computing technique, calculation of leading zeros in a data represented as strings of digits is used very often. Those techniques require high speed of the circuit, as well as its fast design. In this paper we propose a design of such a counter, which is applicable to data length of $w = 4j$ bits, for $4 < j \leq 8$. With this solution it is also possible to process longer data, since the suggested technique offers a good modularity. This is very important, considering the current technology scaling trends. In this paper, a delay behavior of the proposed circuit has also been investigated using equations and VHDL simulation based worst-case delay estimation method. The results show a significant improvement of the circuit speed, compared to the known solutions.

K e y w o r d s: leading zeros count, all zeros detection, modular design

1 INTRODUCTION

In computer technique, it is often necessary to count the leading zeros or ones in a data represented as strings of binary digits. Normalization of the significand in the floating point arithmetic is maybe the most illustrative example. These operations can also be used in the encoding of an anticipator for leading zeros in the floating point arithmetic, as well as for some fixed point dividing algorithms. Certain types of processors (such as the MIPS family) have special instructions in their sets for counting leading ones and leading zeros in integer data.

Detection of leading zeros and/or ones, or their counting has been considered separately [1, 5, 6, 7] or in a framework of leading zeros anticipation [2, 3, 4]. The solution proposed here can make an improvement in two ways. First, we obtain a network with lower number of logic levels, and thus lower propagation time, and second, the proposed solution allows efficient upscaling for longer data using 32-bit counters as building modules.

2 LEADING ZEROS/ONES COUNTER AND ITS SYNTHESIS

Number of leading zeros is defined as the number of zero digits in the most significant positions of data, up to the position in which the first one is present. Analogously, leading ones are ones in the most significant positions of data, up to the position in which the first zero appears. These definitions may by presented in the forms of 0^m1x* for m leading zeros, and 1^m0x* for m leading ones, where x is either 0 or 1, the superscripts represent m instances of the digit 0 or 1, and * represents zero or more instances of the digit x. For example, in the binary datum $00001xxx\dots xxx$ the count of leading zeros is four, and in the datum $110xxx\dots xxx$ the count of leading ones is two.

We will consider 32 bit long data, but the solution that we present is generally applicable to data length $w = 4j$ bits, for $4 < j \leq 8$. By simply copying the solution (with minor modifications) as a building block of a larger counting circuit, much longer data can be processed. Let us first divide the 32 bit data $X(31:0)$ to 4 bits nibbles.

X:	$31\div28$	$27\div24$	$23\div20$	$19\div16$
	a_0, Z_0	a_1, Z_1	a_2, Z_2	a_3, Z_3

X:	$15\div12$	$11\div8$	$7\div4$	$3\div0$
	a_4, Z_4	a_5, Z_5	a_6, Z_6	a_7, Z_7

In this data the 31-st bit is the Most Significant Bit (MSB) and the bit 0 is the Least Significant Bit (LSB). The nibble that contains the MSB is nibble 0, while the nibble that contains the LSB is nibble 7. Labels below the nibbles, a_i, $0 \leq i \leq 7$, denote complements of logical sums (NOR) of data bits in the ith nibble, and Z_i denotes the count of leading zeros in that nibble.

General expression for a_i is

$$a_i = \overline{x_{31-4i} + x_{31-(4i+1)} + x_{31-(4i+2)} + x_{31-(4i+3)}} \quad (1)$$

To simplify the notation of the bits labels in each nibble, we introduce the index $k = 31 - 4i, i = 0, \dots 7$. Having this in mind, by the transformation of (1) into logic product of complements of data bits, we get

$$a_i = \overline{x_k} \cdot \overline{x_{k-1}} \cdot \overline{x_{k-2}} \cdot \overline{x_{k-3}} \quad (2)$$

In (1) and (2) when $a_i = 1$, it means that all bits in the i-th nibble are zeros, otherwise some of them are

* Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia, nebojsa.milenkovic@elfak.ni.ac.rs, vladimir.stankovic@elfak.ni.ac.rs, miljana.milic@elfak.ni.ac.rs

DOI: 10.2478/jee-2015-0054, Print ISSN 1335-3632, On-line ISSN 1339-309X © 2015 FEI STU

ones. The nibble with the smallest value i with $a_i = 0$ is the nibble in which the continuous string of leading zeros terminates, followed by the first one. We will call such a nibble a boundary nibble. If the ith nibble is a boundary nibble, then, to count the leading zeros, the following equations should be satisfied

$$a_0\, a_1\, \ldots\, a_{i-1} = 1 \tag{3}$$

$$a_0\, a_1\, \ldots\, a_{i-1} \cdot \bar{a}_i = 1 \tag{4}$$

The number of leading zeros can be found as a sum of the values $4n$ and Z_n, where $0 \le n \le 7$ is the ordinal number of the boundary nibble with one or more ones, and Zn is the number of terminal zeros in this boundary nibble. For example, in the datum 0000 0000 0000 0011 1001 1111 0111 1000, nibbles with ordinal numbers $0 \div 2$ contain only zeros, while the nibble with ordinal number 3, beside 2 zeros also contains 2 ones. That is why the nibble with ordinal number 3 is the boundary nibble, with 2 terminal zeros, thence $n = 3$ and $Z_n = 2$, and the number of leading zeros is $m = 4n + Z_n = 14$.

The total number of leading zeros can be obtained by concatenation (||) of the three bits of n and two bits of Z_n, since the values of n (in the range $0 \div 7$) are multiplied by 4, while Z_n can have values in the range $0 \div 3$, so $m = n||Z_n$.

If we represent the binary value of n as logic functions y_2, y_1, and y_0 we can get the following expressions

$$Q = a_0 \cdot a_1 \cdot a_2 \cdot a_3 \cdot a_4 \cdot a_5 \cdot a_6 \cdot a_7 \tag{5}$$

$$y_2 = a_0 \cdot a_1 \cdot a_2 \cdot a_3 \tag{6}$$

$$y_1 = a_0 \cdot a_1 \cdot (\bar{a}_2 + \bar{a}_3 + a_4 \cdot a_5) \tag{7}$$

$$y_0 = a_0 \cdot (\bar{a}_1 + a_2 \cdot \bar{a}_3) + (a_0 \cdot a_2 \cdot a_4 \cdot (\bar{a}_5 + a_6) \tag{8}$$

Here $Q = 0$ signs that y_2, y_1, and y_0 are valid, otherwise $(Q = 1)$ they are undefined.

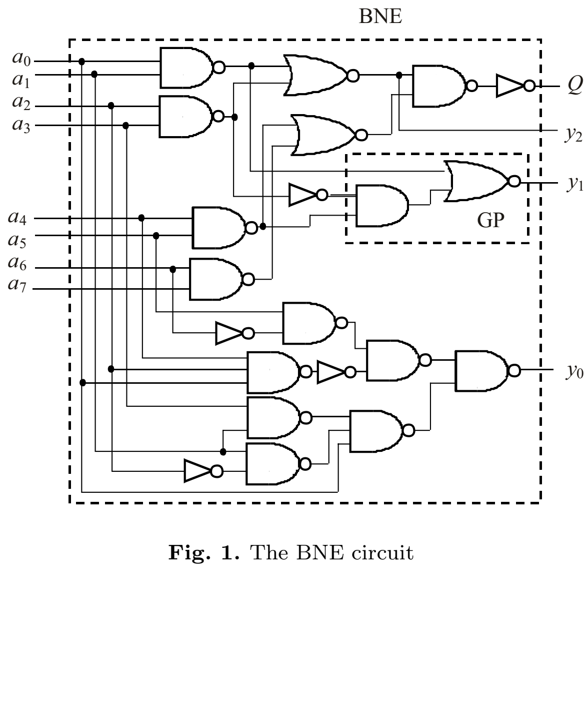

Fig. 1. The BNE circuit

We wanted to develop a logic network to implement (5) to (8) with a minimal propagation delay. We used NOT, NAND and NOR gates with a few additional GP (AND-NOR) gates, where necessary. We omit the proper expressions, for simplicity. The result is shown in Fig. 1, as the BNE (Boundary Nibble Encoder) circuit. The outputs y_2, y_1 and y_0 directly represent the bits $4 \div 2$ of the leading zeros count. The flag $Q = 1$ indicates that the leading zeros count is equal to the data length, in this case, 32. Otherwise $Q = 0$.

We also need to implement 8 networks that will generate a_0, Z_0, a_1, Z_1, $\ldots$, a_7, Z_7. We named them the Nibble Local Count (NLC) networks; one of them is shown in Fig. 2. Here $Z_i = \{z_1^i\, z_0^i\}$ is the binary representation of the leading zeros number of the i-th nibble.

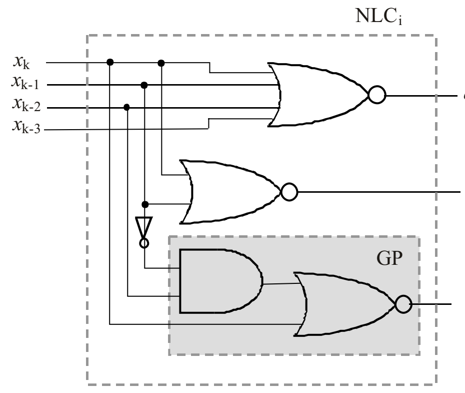

Fig. 2. The NLCi network

Finally, Fig. 3 depicts the complete block diagram of the leading zeros counter. The bold line on the leading zeros counters output represents grouping of the three output lines from the BNE, which carry bits 4:2, and the two output lines from the MUX, which carry bits 1:0, thus defining five bits of the leading zeros count in the range $0 \div 31$. The output line Q from the BNE network, indicates that, when equal to 1, all 32 data bits are zeros. In that case, 4:0 bits of the counters output should be ignored.

The proposed 32-bit leading zeros counter can be adapted for smaller data length $w = 4j$, $4 < j < 8$, for example let $w = 24$ bits, $j = 6$ nibbles. First, we have to ensure the correct response when the leading zeros count is equal to the data length. To provide that $Q = 1$ we must check whether the register X contains only zeroes in the bit positions $X(7:0)$. Then we have $a_6 = a_7 = 1$, which makes the networks NLC6 and NLC7 unnecessary, and we can only apply ones to the BNEs inputs a_6 and a_7 instead. Multiplexers should also be reduced considering their number of inputs (should now be $3 \rtimes 6/1$ type instead of $3 \times 8/1$ type). The BNE circuit does not need any changes. Additionally, this circuit may be reduced by removing unnecessary gates or gates inputs along the paths from inputs a_6 and a_7, toward its outputs.

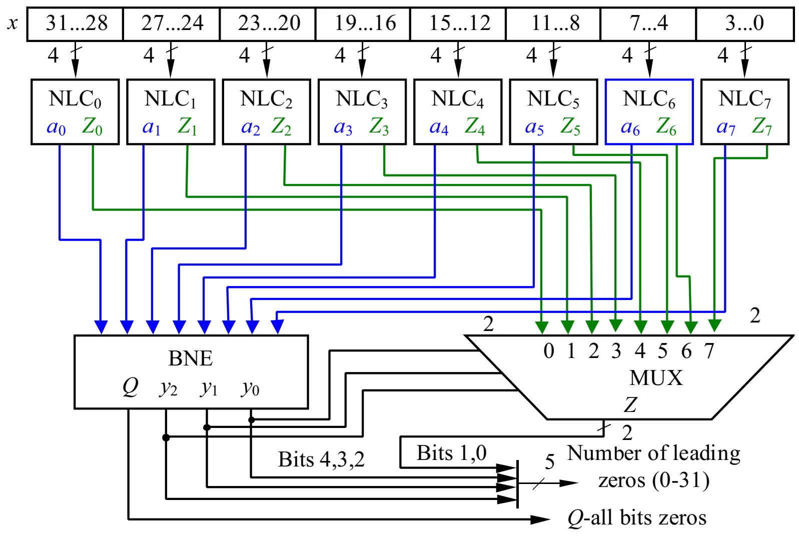

Fig. 3. Architecture of the leading zeros counter for 32-bit data

3 EXTENDING THE SOLUTION FOR LONGER DATA

This solution can be extended to count leading zeros in longer data, for example 64 bits, as follows. Let the network from Fig. 3 with outputs Number of leading zeros (0-31) and Q be one module with outputs NLZ (bits 4:0) and Q. We can simply connect two such modules, as shown in Fig. 4, with two Q outputs: Q_H (higher) and Q_L (lower), as well as two NLZ outputs (NLZH and NLZL). The signal $Q_H = 0$ shows that data bits 63 to 32 contain not only zeros, and the count of leading zeros in the range $0 \div 31$ is determined by $0||NLZH$. Here 0 is the most significant bit, and NLZH gives five lower bits in the six bits count of leading zeros. When $Q_H = 1$ and $Q_L = 0$, the most significant 32 data bits are all zeros, and the 32 lower data bits contain zeros and ones. The number of leading zeros is in the range $32 \div 63$, and is determined by $1||NLZL$. Finally, for $Q_H = 1$ and $Q_L = 1$ all data bits are zeros.

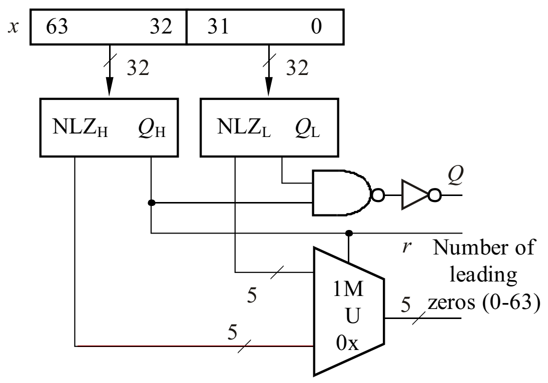

Fig. 4. Block scheme of the leading zeros counter for 64-bit data

In the same way we can make a leading zeros counter for 128 bit long data, using 4 modules, as shown in Fig. 5.

4 PERFORMANCE ANALYSIS

Considered as a building block of the existing processor microarchitecture, the proposed leading zeros counter should meet a certain level of timing performances. In the performance evaluation we have considered worst-cases of time delays through all the paths of the counter. We chose to implement the proposed circuits in a standard CMOS technology from TSMC 65nm Core Library [9].

To verify the logic function and the timing specifications of the logic circuit, logic simulators that consider gate level descriptions are used. However, the delay of the circuit obtained by a standard logic simulation process depends on the applied input test vectors. Determining maximal and minimal delays for all the paths in a combinational circuit, requires 2^n (n is the number of primary inputs in a circuit) simulations using all possible combinations of the input vectors. Therefore, the simple logic simulation is not an efficient timing analysis solution.

Using the data from [9], we have performed a simulation based timing analysis [10, 11, 12] in VHDL by Aldecs Active-HDL. In this analysis, a suitable extension of the logic simulation process enables all the worst case delays for all the paths in one digital circuit to be obtained with only one run of the simulator. To achieve high accuracy, specific gate delay modeling is necessary. With this technique of early timing performance analysis, circuit performance characterization can be made in the first design phases, and thus early timing problems can be prevented and corrected. Accurate path delays in the circuit can be obtained in the final steps of the design process, when the circuit delays are calculated or measured after the synthesis of the layout. If such delays do not satisfy the required timing performances, the circuit needs a redesign. The very action is expected by the occurrence of the timing problems. This encourages the timing performance evaluation to be done in the early phases of the circuit design.

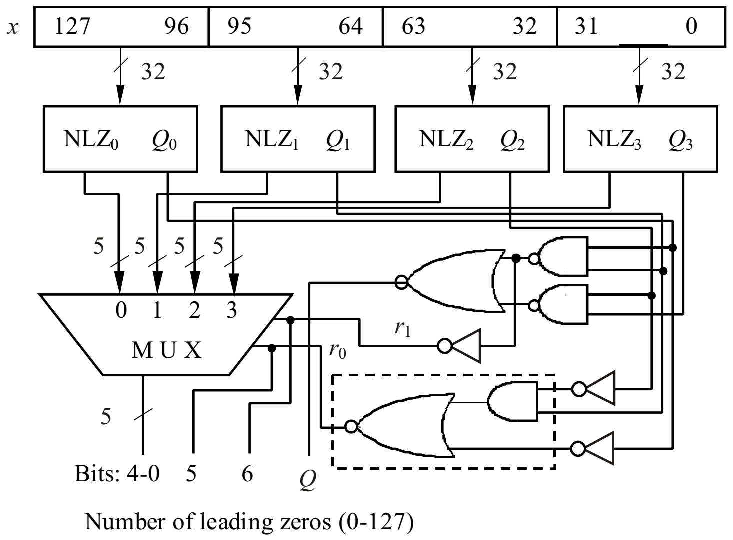

Fig. 5. Block scheme of the leading zeros counter for 128-bit data

On the other hand the aim of such an analysis is to determine how much time might be needed to ensure that the circuit response to any input vector would always occur within that time. It is usually impossible to establish the strict worst-case circumstances. Instead, a scenario, that is almost that bad, is assumed. When we analyze a digital circuit by considering the maximal number of gates on structural paths for instance, we can determine the longest possible path through the circuit, even if an exact stimulus that could sensitize this path is not possible. Even more, such an input or input vector may not be real. Nevertheless, designers never give up analysis of such false paths, since they can, although very rarely, be activated by some event which is independent from the stimulus. Those are the most important motives for preferring this timing performance evaluation method.

Though, in order to calculate four different worst case delays for all the paths in one digital circuit, the gate models must contain all four types of the delays. It means that each gate in a circuit is characterized with four parameters: the minimal delay of the rising edge TPLH, the maximal delay of the rising edge TPLH, the minimal delay of the falling edge TPHL and the maximal delay of the falling edge TPHL.

The obtained propagation times for all the paths in the proposed leading zeros counting circuit are shown in Table 1. From this table we can notice that the minimal propagation times TPHL and TPLH for the six output lines do not differ significantly ($< 40\,\%$ for both of them). The situation is opposite for the maximal values of TPHL and TPLH times, where these times for 2 LSB outputs - bit0 and bit1 are 1.9 to 4.5 times larger than the proper times for outputs y_2, y_1 and y_0. This suggests that the multiplexers in the path for forming the bit0 and bit1 signals are critical components considering their influence on the propagation times along the paths. Better implementation of the multiplexer, such as a pass-transistors logic, can significantly reduce both maximal TPHL and TPLH times.

Since the circuit in Fig. 4 from [7] represents one of the latest and currently fastest solutions for counting leading zeros, we have compared its version extended for 32 bit data with our solution, using the same implementation technology. By simulating this solution under the same conditions as ours, we have derived results presented in Table 2. Significant differences between the minimal and maximal propagation times for both TPHL and TPLH can be noticed from Tables 1 and 2.

Table 1. Worst case propagation times for the output lines of the circuit in Fig. 3

	T_{PHL} (ns)	T_{PLH} (ns)
bit0	$0.125 \div 0.408$	$0.155 \div 0.398$
bit1	$0.125 \div 0.408$	$0.147 \div 0.393$
y_0	$0.084 \div 0.177$	$0.101 \div 0.207$
y_1	$0.076 \div 0.211$	$0.139 \div 0.191$
y_2	$0.079 \div 0.090$	$0.159 \div 0.195$
Q	$0.111 \div 0.143$	$0.162 \div 0.256$

Table 2. Worst case propagation times for the output lines of the circuit in Fig. 4, from [7]

	T_{PHL} (ns)	T_{PLH} (ns)
$\bar{Z}_0$	$0.273 \div 0.310$	$0.172 \div 0.197$
$\bar{Z}_1$	$0.230 \div 0.502$	$0.153 \div 0.414$
$\bar{Z}_2$	$0.231 \div 0.428$	$0.154 \div 0.379$
$\bar{Z}_3$	$0.228 \div 0.366$	$0.150 \div 0.344$
$\bar{Z}_4$	$0.229 \div 0.312$	$0.151 \div 0.309$
$\bar{V}$	$0.228 \div 0.252$	$0.150 \div 0.163$

The reason for this lies in the significant difference of the propagation times between the different inputs of the same gate. This kind of delay estimation sometimes

Table 3. Comparison of the propagation times for both circuits, expressed as percentage delay decrease for all delay types and all paths

	T_{PHL} min (%)	T_{PHL} max (%)	T_{PLH} min (%)	T_{PLH} max (%)
bit0 *vs* $\bar{Z}_0$	-54.21	31.61	-9.88	102.03
bit1 *vs* $\bar{Z}_1$	-45.65	-18.73	-3.92	-5.07
y_0 *vs* $\bar{Z}_2$	-63.64	-58.64	-34.42	-45.38
y_1 *vs* $\bar{Z}_3$	-66.67	-42.35	-7.33	-44.48
y_2 *vs* $\bar{Z}_4$	-65.50	-71.15	5.30	-36.89
Q *vs* $\bar{V}$	-51.32	-43.25	8.00	57.06

makes the propagation time of the longest path to be several times greater than the time for the shortest path in the circuit. As such, the obtained propagation times are the absolute theoretical minimal and maximal propagation times in the circuit. These are the results of the timing analysis of all the paths in the circuit from the topological point of view, and not the timing analysis of the paths that we want the data to go through. Obviously, the real propagation times will be somewhere within the obtained ranges. Comparison of the maximal values for both TPHL and TPLH in Tables 1 and 2 shows that the proposed solution has smaller propagation times, especially for the higher weight counters bits. This fact makes the proposed solution especially suitable in the shift register control during the results normalization in floating point operations. Also, if we compare these two solutions considering the total maximal delay for all the outputs, our solution is better for both T_{PHL} (0.408 *vs* 0.502) and T_{PLH} (0.398 *vs* 0.414).

Table 3 shows improvements of the worst-case delays of our solution. Here we have used the formula $((T_{PS} - T_{S7})/T_{S7}) \times 100$ (in %), where PS stands for proposed solution and S7 for solution from [7]. This formula basically calculates decreases in the delay of our solution compared to the solution from [7], which means that the minus sign actually denotes an improvement. The results are rather good, since -50% practically means twice faster, -67% three times faster *etc*, while +100% means twice slower (*ie* the first solution is twice faster). Similar results in the delay analysis were obtained using the timing evaluation based on Logical effort theory [8].

At the end, it should also be mentioned that the proposed technique for counting zeros uses about 10% less number of basic logic gates for the same operation which consequently leads to less area occupation in the integrated circuit.

5 CONCLUSION

A design methodology for a leading zeros counter is presented in this paper. The proposed design represents a basic module for 32-bit data, while the used methodology is also easily applicable to a design of leading zeros counter for data length of $4j$ bits, for $4 < j \leq 8$. By adding simple logic to multiple copies of the basic module, we can easily process longer data, like 64 bit, 128 bit, or more.

Major advantage of this leading zeros counter are smaller delays along the circuit paths comparing to the similar circuits, for the 0.65 μm technology, which makes it faster and more reliable in time. Another improvement of the proposed solution is a relatively smaller hardware requirement. To implement the circuit blocks (NLCi, BNE, MUX), basic logic gates are used. This methodology also enables a simple implementation of the circuit modules to achieve upscaling for longer data, with small delay increase.

References

[1] OKLOBDZIJA, V. G.: An Algorithmic and Novel Design of a Leading Zero Detector Circuit: Comparison with Logic Synthesis, IEEE Trans. VLSI Systems **2** No. 1 (March 1994), 124–128.

[2] SUZUKI, H.—MORINAKA, H.—MAKINO, H—NAKASE, Y. MASHIKO, K — SUMI, T.: Leading-zero anticipatory logic for high-speed floating point addition, IEICE Journal of Solid-State Circuits **31** No. 8 (August 1996), 1157–1164.

[3] SCHMOOKLER, M. S.—NOVKA, K. J.: "Leading Zero Anticipation and Detection - A Comparison of Methods, Proc. 15th IEEE Symposium on Computer Arithmetic (2001), 7–12.

[4] ARAKAWA, F. Hayashi,—T.—NISHIBORI, M.: An Exact Leading Non-Zero Detector for a Floating-Point Unit, IEICE Trans. Electron. **E88-C** No. 4 (April 2005), 570–575.

[5] OTT, M.: Optimized Method and Aparatus for Parallel Leading Zero/One Detection, US Patent #6697828B1 (February 2004).

[6] YIN, S.-H.—GOWAN, M. K.: High Speed Leading or Trailing Bit Value Detection, US Patent #7012965B2 (March 2006).

[7] DIMITRAKOPOULOSG.—GALANOPOULOS, K.—MAVROKEFALIDIS, C.—NIKOLOS, D.: Low-Power Leading-Zero Counting and Anticipation Logic for High-Speed Floating Point Units, IEEE Trans. On VLSI Systems **16** No. 7 (July 2008), 837–850.

[8] WESTE, N.—HARRIS, D.: CMOS VLSI Design: A Circuit and System Perspective, third ed., Boston, MA Addison Wesley, 2005.

[9] TCBN65LP TSMC 65nm Cole Library Data Book, version 1.4, September 2008.

[10] SOKOLOVIC, M.—LITOVSKI, V.: Using VHDL simulator to estimate logic path delays in combinational and embedded sequential circuits, Proc. IEEE region 8 Eurocon 2005 Conference, Belgrade (2005), 547–550.

[11] M. SOKOLOVIC, V. LITOVSKI—M. ZWOLINSKI: New Concepts of Worst-Case Delay and Yield Estimation in Asynchronous VLSI Circuits, Microelectronics Reliability, Volume 49, Issue 2, pp. 186-198, February 2009.

[12] SOKOLOVIC, M.—LITOVSKI, V—ZWOLINSKI, M.: Efficient and Realistic Statistical Worst Case Delay Computation Using VHDL, Electrical Engineering (Archive fur Elektrotechnik), **91** No. 4-5, 197–210, online: November 2009.

Received 3 July 2014