Statistical Timing Analysis of Asynchronous Circuits Using Logic Simulator

Miljana Lj. Sokolović and Vančo B. Litovski

Abstract—**The lack of methods and tools for performance estimations in asynchronous circuits is one of the main reasons why this design methodology, beside its advantages, is still unpopular among designers. Using a logic simulator it is possible to efficiently estimate all worst-case path delays in one asynchronous circuit, which can be crucial for overcoming this problem. This paper describes a method for statistical estimation of topological delays in asynchronous circuits, based on the application of a VHDL simulator. The method is verified on a set of chosen asynchronous circuits and in compare with other similar methods shows higher efficiency.**

Index Terms— **Asynchronous logic circuits, Logic simulator, Timing analysis.**

I. Introduction

ASYNCHRONOUS integrated circuits design style is the one that designers rather avoid, beside its unquestionable advantages. Asynchronous circuits don't have the clock signal, and the problems related to clock routing and distribution, such as the clock skew are avoided. The absence of the clock lines gives a significant size reduction for the integrated circuit. These circuits are characterized with a good modularity, easier technology migration and down scaling. The energy is consumed only while the useful work is done. The absence of the clock signal also significantly contributes to the reduction of the power consumption. These circuits have less EMI levels (and though the decreased emanation from the chip, which make the side-channel attacks easier), and are more resistant to noise. All those advantages are very important, especially while designing mobile systems where the battery size and its duration are the key factors [1].

Nevertheless, the absence of the clock signal means that the events in the asynchronous circuits cannot be accurately predicted, as it can be with the synchronous circuit, which is the mail reason for their poor design tools support. Also, synchronous circuits have a larger commercial practice [2].

Manuscript received April 2, 2009.

M. Lj. Sokolović is with the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (phone: +381-18-529-321; fax: +381-18-588-447; e-mail: miljana.sokolovic@elfak.ni.ac.rs).

V. B. Litovski, is with the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: vanco.litovski@elfak.ni.ac.rs).

All these issues give a weak motivation for the usage of this design style.

One of the problems related to the asynchronous circuit design technique that is still unsolved is the estimation of their performances. In other words, it is necessary to determine the delays of all paths in one asynchronous circuit. This estimation, performed in the earliest design stages, would significantly contribute to early detection of the bad design solutions, and would at the same time be used as a tool for the early estimation of the operation speed for the new designed circuit. More accurate delays could be determined in the final design steps, after the circuit layout synthesis. Though, if the obtained circuit operating speed is not satisfactory after all these steps, or a particular timing problem occurred, the circuit has to be redesigned, and the design brought back to the beginning. At the end, one can come to a conclusion that the performance estimation is the best to be performed in the early design stages, that is, right after the first circuit description and the simulation [3].

The simulation is the simplest way to determine a delay in a circuit. But, it is very inefficient to simulate large circuit at the transistor level of abstraction. Logic simulators use simplified gate models and significantly speed up this process. Using the logic simulations, it is possible to verify the logic function as well as the behavior of the circuit in the observed time period. Nevertheless, the delays obtained in this way depend on the applied combination of the input vector. In order to determine the largest and the smallest delays for a circuit, it has to be simulated for all possible 2^n input vector combinations, where n represents the number of circuit inputs. For the circuits with a large number of inputs this approach is also inefficient. At the other hand, since the logic simulator is used for the first design stages, implementing the method for worst-case delay estimation into the standard logic simulator, would ensure the early detection of the incorrect design solutions.

The second aspect that strongly affects the obtained integrated circuit yield is the tolerance of the technology. Even with the perfectly designed circuit, it could often happen that the use of more tolerant processes, gives many circuit that do not satisfy the required timings. In other words, we produce a circuit whose response is outside the acceptable limits. Parameters of the particular electronic circuit have the statistical distribution within a particular interval. Logic simulator which is enhanced with the procedures for worst-case delay estimation, is capable to simulate these phenomena, in order to achieve more accurate estimation results, used in final yield estimation.

To achieve more accurate delay estimation results, another aspect must be taken into account, and that is the circuit implementation. The fanout value of each gate in the circuit netlist could be very important, but is often a neglected factor in the logic behavior and timing analysis. This fact brings one to a conclusion that this factor must be taken into account while implementing the method for a digital circuit path delay estimation.

In this paper we will try to demonstrate the application of the standard logic simulator in the worst-case delay estimation for all paths of the asynchronous circuit. The statistical delay estimation method that is suggested in this paper takes into account both the tolerances of the technology and the circuit implementation, while using a very accurate models of the gates' timing behavior. The advantages of this method are its simplicity, efficiency and the ability to be implemented it into any standard logic simulator. The following paragraphs will give the description of the method, its implementation into a VHDL simulator, and the procedures for statistical processing of the obtained results using the Matlab program.

II. Asynchronous Circuit Delay Estimation Using a Logic Simulator

In order to enable timing analysis using a standard logic simulator, it is necessary that signal and gate descriptions carry the information about delays, while the signal logic values become irrelevant in this case. Signals should be described with a specific attributes which define events and delays. Signals described in this way activate processes inside the gates and change the current values of the signal attributes. In order to achieve the large speed of such an analysis, the simultaneous propagation of all possible input vector combinations through the circuit is assumed. Nevertheless, it does not mean that the circuit has to be simulated for every possible input vector combination. Instead of it, with one analysis run all possible input vector combinations are analyzed for each gate in the circuit and only those considered as worst-cases sent further through the circuit until the primary outputs of the entire circuit are reached. The delay values are accumulated along the paths in the circuit starting from the primary inputs, and ending at the primary outputs, or some other point inside the analyzed circuit. At the end of this, very fast process, the largest and the smallest delay values for the rising and the falling edges of all output signals are available.

For each signal S in the circuit, four delay types are estimated::

--d1mn(S) – the shortest path delay for a rising edge at S,

--d0mn(S) – the shortest path delay for a falling edge at S,

-- d1mx(S) – the longest path delay for a rising edge at S, and

-- d0mx(S) – the longest path delay for a falling edge at S.

In order to enable simultaneous propagation of all possible input vector combinations, and to enable the calculation of all mentioned delay values, signals that connect gates inside the circuit must carry two types of information. They are represented in a form of two types of attributes: attributes that carry the information about signal events, which initiate the calculation processes inside the gates, and the attributes that contain the information about all listed delay types.

Processes inside the gates caa process signals described in this way, and it requires two-modes gate models, that is: activation-propagation mode and the delay calculation mode. Also, in order to enable the calculation of all delay types, the description of a gate model must contain two processes: one for the calculation of the maximal delay for the rising and falling signal edges, and one for the calculation of the minimal delay for the rising and falling signal edges. The activation-propagation mode of the gate model in each of these processes is sensitive to any change of the attribute for initiation of the delay calculation. When this mode is activated, an output signal is given new value of the delay attribute, taking into account the values of the delay attributes for all gate input signals, and the particular delay value of the observed gate. When this value is updated, the initiating attribute of the gate output signal is also changed in order to initiate the delay calculations in the gates that topologically follow.

The basic principle of the delay accumulation process is described in figure 1. The figure illustrates the calculation of the maximal delays along all paths of one tree-input C element. In this case, both rising and falling edges are applied at all circuit inputs. Inside each gate, new delay values for rising and falling edges are obtained. The delay estimation process ends when all these transitions reach the circuit primary outputs. In this case the delay analysis along all the paths in the circuit is possible only after the feedback line is broken. It makes sense to analyze the delays along the paths of the circuit in only one operating sequence.

Nevertheless, it could also be interesting to determine the maximal delay along the signal paths that go through the feedback line. In this way it is possible to analyze the timing behavior for more operating sequences of the circuit. The suggested delay estimation method could be extended to analyze such cases, if one applies the principle similar to one used for sequential circuit test vector generation [4]. It means that in order to analyze timing of the few operating sequences in a circuit, that circuit should be replicated and analyzed that many times. Figure 2 illustrates the application of such a principle for estimating the delay of the longest path for an asymmetrical C-element circuit. It is also necessary to break the feedback, as shown in the figure. From the

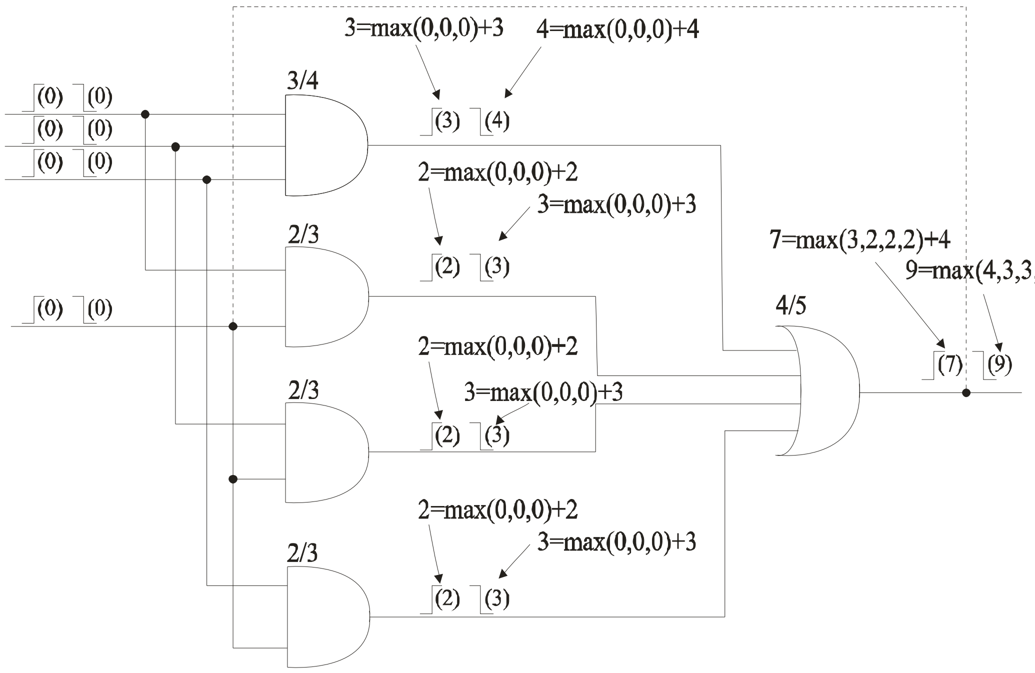

Fig. 1. The estimation of the maximal delay for a tree-input C element

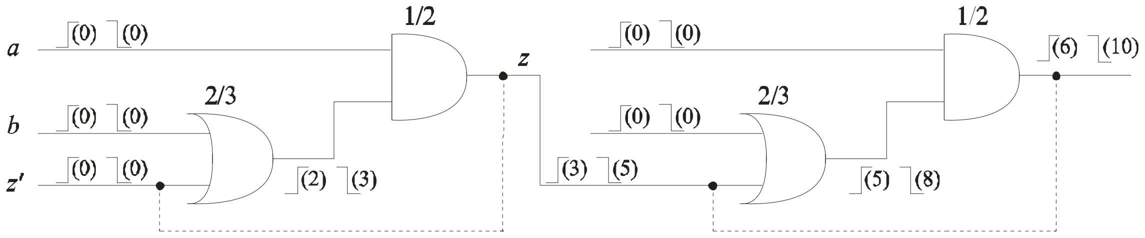

Fig. 2. Maximum delay estimation through the sequences

implementation point of view, the circuit netlist does not have to be rewritten few times. Instead, the results of the delay estimation processes should be applied to the input of the circuit that makes the feedback. After that, the circuit should be analyzed again, with a new initial delay parameter values.

Each gate is described with four types of delay. They are maximal delay of the rising edge through the gate, minimal delay of the rising edge through the gate, maximal delay of the falling edge through the gate and the minimal delay of the falling signal edge. Although these values are fixed for each gate, the delay assignment process in each gate is much more complex. Two factors determine the gate delay value. First, the real implementation of the circuit is taken into account and the particular gate position within the circuit netlist. It means that gate's fanout affects the gate delay and this dependence is expressed with a specific function. The second factor is related to the gate's initial delay value, and this value is fixed. But, since our intention is the statistical delay analysis and the estimation of the process tolerances influence, it is necessary to randomly generate the delay value, which is defined with its mean and deviation according to the given delay distribution. The mean represents the fixed chosen, and in our case is set to 3% value. The Gaussian delay distribution function is applied. Whenever the delay calculation process is initiated inside a gate, a special function generates the random value [3].

Statistically satisfying estimation results can be obtained after few hundred analyses. This fact should not intimidate since these analysis require very little time. The exact number of simulations is determined with a required result precision. The circuit also has to be described at the structural level. At the beginning of the analysis, both rising and falling edges are simultaneously applied to all circuit inputs. All these events at the circuit inputs initiate the delay calculation processes in gates from the first topological level of the circuit. When these processes terminate, the delay attribute values of the gates' output signals can be updated, and the activation-propagation attribute values change in order to enable the delay calculation processes in the gates from the second topological level. This wave of calculation is moving from the primary inputs until the primary outputs are reached. The analysis then terminates, and it gives as a result, all delay attributes for all output signals in the circuit.

III. VHDL Implementation

As already mentioned, suggested concept is implemented in VHDL environment and analyses are done using standard VHDL simulator. The best way to compute numerous results obtained through the statistical analyzes is automatically, so for that purpose Matlab software package is used. This software contains integrated procedures for mean value and deviation calculation under the set of numerous samples, as well as the tools for drawing histograms for circuits with small number of outputs.

```
p1: process (in1.d0mn, in1.d1mn, in1.arr0mn, in1.arr1mn,
        in2.d0mn, in2.d1mn, in2.arr0mn, in2.arr1mn)
        variable r,p: real;
        variable multipl : real;
begin
        multipl := real(ifo_izl);
        f<=fanout_func(multipl)
        r:= ((f*1.0) + (0.03*(gauss_rng)));
        p:= ((f*0.9 + (0.03*(gauss_rng)));
        if (in1.arr0mn and in2.arr0mn ) then
                out1.d0mn   <= min(in1.d0mn, in2.d0mn) + r;
                out1.arr0mn <= true;
        end if;
        if (in1.arr1mn and in2.arr1mn) then
                out1.d1mn   <= min(in1.d1mn, in2.d1mn) + p;
                out1.arr1mn<= true;
        end if;
end process p1;
p2: process (in1.d0mx, in1.d1mx, in1.arr0mx, in1.arr1mx,
        in2.d0mx, in2.d1mx, in2.arr0mx, in2.arr1mx)
        variable r,p: real;
        variable multipl : real;
begin
        multipl := real(ifo_izl);
        r:= (multipl*0.95 + (0.03*(gauss_rng)));
        p:= ((multipl*1.05) + (0.03*(gauss_rng)));
        if (in1.arr0mx and in2.arr0mx) then
                out1.d0mx   <= max(in1.d0mx, in2.d0mx) + r;
                out1.arr0mx <= true;
        end if;
        if (in1.arr1mx and in2.arr1mx) then
                out1.d1mx   <= max(in1.d1mx, in2.d1mx) + p;
                out1.arr1mx<= true;
        end if;
end process p2;
```

Fig. 3. VHDL implementation of the processes within the two-input C-element

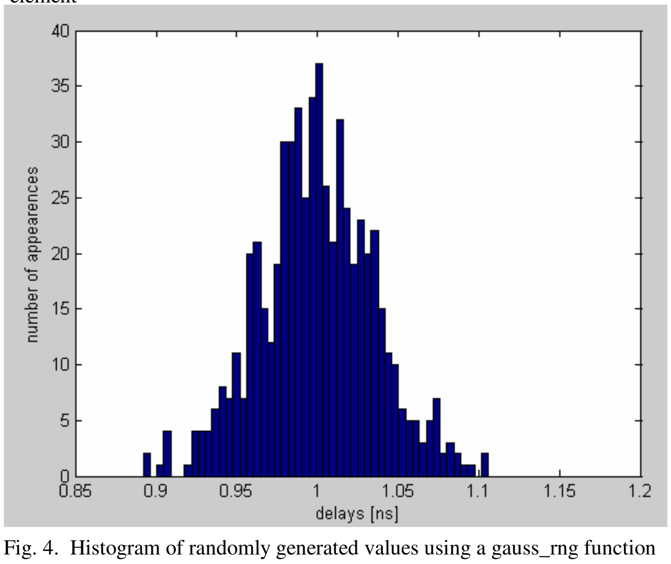

Fig. 4. Histogram of randomly generated values using a gauss_rng function

VHDL models of all gates and simple asynchronous elements are kept in a particularly developed library. Figure 3 shows the implementation of a two-input C-element. This description contains a numerous calls of gauss_rng function. This function, for a given mean value and deviation, randomly generates numbers with Gaussian distribution. In order to verify this function, a test environment is built under which this function was executed 600 times, with the adequate parameters sets. The results are processed using Matlab and the corresponding histogram is obtained, and given in figure 4, showing the excellent results.

```
entity RSLatch is
    generic (ifo_izl_1: integer:= 1;
        ifo_izl_2: integer:= 1;
        tr_rq_mn : real := 1.0e-9;
        tf_rq_mn : real := 0.9e-9;
        tr_rq_mx : real := 1.05e-9;
        tf_rq_mx : real := 0.95e-9;
        tr_rnq_mn : real := 1.0e-9;
        tf_rnq_mn : real := 0.9e-9;
        tr_rnq_mx : real := 1.05e-9;
        tf_rnq_mx : real := 0.95e-9;
        tr_sq_mn : real := 1.0e-9;
        tf_sq_mn : real := 0.9e-9;
        tr_sq_mx : real := 1.05e-9;
        tf_sq_mx : real := 0.95e-9;
        tr_snq_mn : real := 1.0e-9;
        tf_snq_mn : real := 0.9e-9;
        tr_snq_mx : real := 1.05e-9;
        tf_snq_mx : real := 0.95e-9);
    port (q, nq : out SDA_std_logic := (0.0, 0.0, false, false, 0.0, 0.0,
    false, false);
        r, s : in SDA_std_logic := (0.0, 0.0, false, false, 0.0, 0.0,
    false, false));
end RSLatch;
architecture only of RSLatch is
begin
    p1: process (r.d0mx, r.d1mx, r.arr0mx, r.arr1mx, s.d0mx, s.d1mx,
    s.arr0mx, s.arr1mx)
        variable i, j ,k, l, m, n, o, p : real;
        variable multipl1, mulitipl2 : real;
    begin
        multipl1 := real(ifo_izl1);
        multipl2 := real(ifo_izl2);
        f1<=fanout_func(multipl1);
        f2<=fanout_func(multipl2);

        i:= (f1* tr_rq_mx + (0.03*(gauss_rng)));
        j:= (f1* tf_sq_mx + (0.03*(gauss_rng)));
        k:= (f2* tf_rnq_mx + (0.03*(gauss_rng)));
        l:= (f2* tr_snq_mx + (0.03*(gauss_rng)));
        if (r.arr0mx and s.arr1mx) then
            q.d1mx<=max(r.d0mx,s.d1mx)+max(i, j);
            q.arr1mx <= true;
            nq.d0mx<=max(r.d0mx,s.d1mx)+max(k, l);
            nq.arr0mx <= true;
        end if;

        m:= (f1* tf_rq_mx + (0.03*(gauss_rng)));
        n:= (f1* tr_sq_mx + (0.03*(gauss_rng)));
        o:= (f2* tr_rnq_mx + (0.03*(gauss_rng)));
        p:= (f2* tf_snq_mx + (0.03*(gauss_rng)));
        if (r.arr1mx and s.arr0mx) then
            q.d0mx<=max(r.d1mx,s.d0mx)+max(m, n);
            q.arr0mx <= true;
            nq.d1mx<=max(r.d1mx,s.d0mx)+max(o, p);
            nq.arr1mx <= true;
        end if;
    end process;
end only;
```

Fig. 5. VHDL implementation of the RS-latch circuit – process for determining maximal delay to the circuit output

Figure 5 shows the implementation of the RS-latch circuit. It is assumed that this circuit has two inputs – R and S along with two outputs – Q and NQ. The definition of generics stands at the very beginning of the description. Beside 16 parameters representing the minimal and maximal delays of four possible input – output combinations (R-Q, R-NQ, S-Q, S-NQ) and both possible transitions, the generics description also contains the parameter corresponding to the fanout value ifo_izl. This parameter is initially set to unit value. Inside the netlist description, and during the instantiation of the particular library element, a specially developed program gives the real value to this generic, according to the circuit structure and its topological position [5]. For this particular circuit, two fanout values are required, since the circuit has two outputs. The figure shows the process for determining maximal delays to both outputs. Similar process stands for determining minimal delays.

A unique testbench programs enable the multiple simulations (600) of the analyzed circuit, while writhing the estimation results for all delay types for each circuit outputs into a specific text file. Matlab program reads the corresponding columns in these files and calculates its mean and deviation value for each circuit output. The described results analysis is appropriate for the circuits with a large number of outputs. For the circuit with a small number of outputs, the results can be represented in the form of a histogram.

IV. Results

For a verification of the proposed delay estimation method a set of typical asynchronous circuit is chosen. The problem that occurred here is that there are no asynchronous benchmark circuits, for verification, analysis and comparison of performances for different methods.

TABLE I
Delay Estimation Results for an Asynchronous Counter

Output	Delay type	Topol. level	min/ max	fanout	statistics	
					mean	deviat.
	mnr	4	3.7ns	3.7ns	3.704	0.705
1.	mxr	4	3.9ns	3.9ns	3.898	0.071
	mnf	4	3.6ns	3.6ns	3.599	0.681
	mnf	4	3.8ns	3.8ns	3.799	0.687

Table 1 shows the logic simulator delay analysis results for one asynchronous binary counter containing four T-latch circuit. The first column in the table denotes the number of the output, the second column stands for the delay type of the particular circuit output, while the third shows the topological level for the obtained value of the particular delay type. Following two columns give the worst-case delay analysis results, when fanout value does not and then does affects the delay calculation processes, while the delay generation is not random. The last two columns give the statistical processing of the obtained simulation samples, that is, mean and the deviation value for the particular delay type.

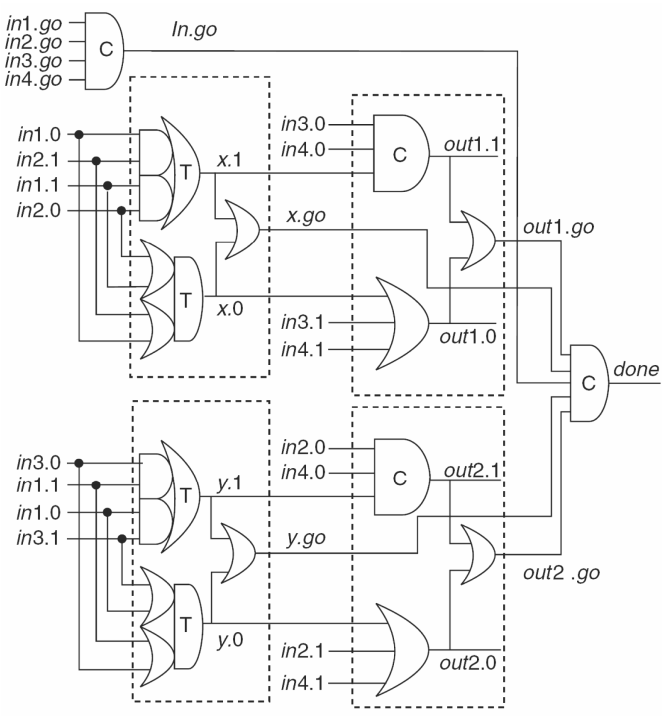

Fig. 6. Asynchronous encoder circuit

A more complex circuit of an asynchronous encoder is shown in figure 6, while the table 2 gives the analysis results for this circuit.

TABLE II
Delay Estimation Results for an Asynchronous Encoder

Output	Delay type	Topol. level	min/ max	fanout	statistics	
					mean	deviat.
	mnr	1	0.9ns	0.9ns	0.900	0.035
1.	mxr	1	0.95ns	0.95ns	0.954	0.036
	mnf	1	1.0ns	1ns	1.000	0.036
	mnf	1	1.05ns	1.05ns	1.051	0.035
	mnr	2	2.0ns	2ns	1.999	0.050
2.	mxr	2	2.1ns	2.1ns	2.101	0.050
	mnf	2	1.8ns	1.8ns	1.801	0.053
	mnf	2	1.9ns	1.9ns	1.905	0.049
	mnr	3	2.8ns	2.8ns	2.773	0.055
3.	mxr	4	4.0ns	4ns	4.000	0.072
	mnf	3	2.9ns	2.9ns	2.898	0.060
	mnf	4	4.0ns	4ns	4.000	0.072
	mnr	2	2.0ns	2ns	1.999	0.052
4.	mxr	3	3.05ns	3.05ns	3.051	0.060
	mnf	2	1.8ns	1.8ns	1.802	0.051
	mnf	3	2.95ns	2.95ns	2.954	0.064
	mnr	1	0.9ns	0.9ns	0.900	0.036
5.	mxr	2	2.0ns	2ns	2.002	0.053
	mnf	1	1.0ns	1ns	1.006	0.035
	mnf	2	2.0ns	2ns	2.002	0.049

Table 3 gives the simulation run times and the corresponding allocated memory for tree different asynchronous circuits: C-element, counter and encoder. The table gives the comparison for two delay analysis concepts: first is the analysis based on the concept described in this paper, while the second is based on the classical application of the standard logic simulations. The last column of this table shows how many simulations are required for a standard logic simulator to simulate all possible input vector combinations. The table illustrates the significantly higher efficiency of the proposed method.

TABLE III
COMPARISON OF TWO DIFFERENT DELAY ANALYSIS APPROACHES

Circuit	Timing analysis		Logic simulation		
	memory allocation [kB]	CPU time [S]	memory allocation [KB]	CPU time [S]	number of simulations
C-elem	6.4	19.734	444	1	16
counter	7.7	20.277	375	2	2
encoder	28.5	92.583	480	436	1048576

V. CONCLUSION

A new concept of statistical worst-case delay estimation in asynchronous circuits is described in the paper. The method is implemented into a standard logic simulator.

Thanks to the specific gate modeling that includes the tolerances of the technology, the circuit specific structure, as well as different delay types that describe the gate's timing behavior, a high reliability of the obtained results is achieved. In compare with a classical Monte-Carlo analysis, the method shows higher efficiency from the allocated memory, and from the duration of the analysis points of view.

REFERENCES

[1] M. Lewis and L. Brackenbury, "CADRE: A Low-power Low-EMI DSP Architecture for Digital Mobile Phones," *VLSI Design*, vol. 3, issue 12, pp. 333-348, 2001.

[2] A. Davis and S. Novick, "An Introduction to Asynchronous Circuit Design," Technical Report UUCS-97-013, Computer Science Department, University of Utah, September 1997.

[3] M. Sokolovic, M. Zwolinski and V. Litovski, "New Concepts of Worst-case Delay Evaluation in Asynchronous VLSI SoC," Proc. 26th International Conference on Microelectronics, MIEL 2008, vol. 2, pp. 377-385, Niš, May 2008.

[4] K. T. Cheng and V. D. Agrawal, *Unified Methods for VLSI Simulation and Test Generation*, Kluwer Academic Publishers, 1989.

[5] M. Sokolović, V. Litovski and M. Zwolinski, "Fan-out Based Delay Estimation in Digital Circuits," *Proc. of the VI symposium on Industrial electronics*, INDEL 2006, pp. 101-104, Banja Luka, Nov. 2006.