Using VHDL Simulator to Estimate Logic Path Delays in Combinational and Embedded Sequential Circuits

Miljana Lj. Sokolović, and Vančo B. Litovski, *Member, IEEE*

Abstract —**This paper presents a VHDL based method that enables the logic simulator to estimate the longest and the shortest path delays of all signals in the circuit with only one run of the logic simulator. The method is verified for ISCAS`85 benchmark circuits and for one particular embedded sequential circuit. It is extremely efficient and appropriate in the early phases of the design process where timing analysis needs to be repeated as the circuit is optimized or redefined.**

Keywords — **delay estimation, simulation.**

I. Introduction

IN order to reduce the design time and to decrease its cost, it is convenient to estimate all path delays in early stages of the design process, possibly during the hardware description. The shortest path delays determine the minimal duration of the clock pulse, while the longest path delays determine the maximal operation circuit frequency. To estimate the minimum clock pulse width and the maximum operating frequency one should determine the primary outputs at which the smallest and the largest delays are exhibited, as well as the ranges of these delays [1], [2]. Hence, one may determine the part(s) of the circuit that is(are) to be redesigned or optimized. Simulators, known as timing analyzers [3] - [5] compute circuit delays by evaluating the response of the circuit to a given restricted set of input patterns. The synthesis of these patterns can be extremely complex. Increasing the number of inputs, however, increases the number of input vectors in an exponential manner. Therefore, the simulation of all possible combinations of input vectors becomes improper and some optimal, vector-independent approach for the worst case delay-path estimation is necessary.

This paper proposes a simple method that enables the logic simulator to quickly estimate worst-case path delays, that is the largest possible ranges of delays at each output of the circuit for both rising and falling edges, with only one run of the logic simulator. Few additional processes are needed to be added into the gates' descriptions. Small improvement of the simulation mechanism is needed, too.

The proposed method is based on the use of composite type of the signals that consist of signal logic value and needed timing data. To test the performance of the proposed method, longest and shortest path delays in ISCAS'85 benchmark circuits [6] are evaluated using a VHDL simulator [7]. The presented results prove an excellent performance of the method.

The paper is organized as follows. In second section the method for tuning logic simulator for timing analysis is described. This method can be applied for combinational and embedded sequential circuits. After that the VHDL implementation of the method is explained. Some small changes into gates' and flip-flops' models are needed, as well as small improvement of the simulation mechanism. This method is verified for ISCAS'85 combinational circuits and, for the first time, for a particular embedded sequential circuit. The results of the simulations are given in the last section of the paper.

II. Estimation Method

When a digital circuit is simulated for one specific input vector, the time instant when the first activity occurs on a signal determines the shortest path delay to that particular signal for the given input vector. The time instant when the last activity occurs on a signal determines the longest path. In order to obtain the worst-case delays of both rising and falling signal transitions the circuit has to be simulated for all input vectors. Therefore, 2^n circuit simulations have to be carried out, where n is the number of primary inputs. It is obvious that this approach is not feasible when n becomes large.

In order to evaluate the longest and the shortest path delays to **all** the signals in the circuit with only **one** simulation of the circuit, simultaneous simulation of the circuit for all input vectors is necessary. To enable logic simulators to perform such a simulation, eight additional signal attributes need to be introduced to store the information about the arrival of the transitions at the signal and of the shortest and the longest path delays to the signal [8]. Having a signal named *S*, these eight attributes are:

d1mn(S) - the shortest path delay of the rising transition at *S*,
d0mn(S) - the shortest path delay of thefalling transition at *S*,

M. Lj. Sokolović is with the Faculty of Electrical Engineering, University of Niš, Serbia & Montenegro (phone: 381-18-529321; fax: 381-18-588399; e-mail: miljana@venus.elfak.ni.yu).

V. B. Litovski is with the Faculty of Electrical Engineering, University of Niš, Serbia & Montenegro (phone: 381-18-529224; fax: 381-18-588399; e-mail: vanco@venus.elfak.ni.yu).

arr1mn(S) - rising transition of the shortest path arrival flag,
arr0mn(S) - falling transition of the shortest path arrival flag,
d1mx(S) - the longest path delay of the rising transition at *S*,
d0mx(S) - the longest path delay of the falling transition at *S*,
arr1mx(S) - rising transition of the longest path arrival flag,
arr0mx(S) - falling transition of the longest path arrival flag.

It is assumed that the circuit is described and simulated at structural level of abstraction and that the delay ranges (minimal and maximal delays) of all building blocks for both rising and falling edge are known. At the start of the simulation the circuit is stimulated with both rising and falling transitions at all primary inputs. As the transitions propagate from primary inputs towards primary outputs, the gate delays are accumulated along the paths. Signal attributes are accessed from the processes in the gate models and their values are dynamically updated. Once the circuit activity is exhausted, the shortest and the longest path delays are available in signal attributes *d1mn, d1mx, d0mn* and *d0mx* of each output signal in the circuit.

The timing simulation mechanism is inserted into gate models with additional processes that monitor and update the values of signal attributes.

For a model of NAND gate the process of maximal delay estimation is represented by the pseudo code shown in Fig. 1.

```
process (a, b) {
  // update falling edge delay of signal f:
  if (arr0(a) .OR. arr0(b))
    d1(f) := MAX(d0(a),d0(b)) + tr;
    arr1(f) := 'true';
    f <= an_event;    }
  // update rising edge delay of signal f:
  if (arr1(a) .AND. arr1(b)) {
    d0(f) := MAX(d1(a),d1(b)) + tf;
    arr0(f) := 'true';
    f <= an_event;    }
}
```

Fig. 1. Two-input NAND gate model for longest path delay estimation with logic simulator.

Gate inputs are denoted *a* and *b*, gate output is denoted *f*, whereas gate propagation delays for rising and falling edges at the output *f* are denoted *tr* and *tf* respectively. Each falling transition at an input of the gate results in rising transition at its output, but rising transition at an input is capable to produce falling transition at the output only if rising transition had previously arrived at the other gate input. Taking into account various input signal slopes, loading capacitances and other parameters that influence ranges of gate delays *tr* and *tf*, one can apply delay models of arbitrary complexity. Each time a delay attribute of output signal *f* is changed, an event must be scheduled to signal *f* with zero delay in order to activate the processes performing the timing simulation in the gates driven by the NAND gate.

The example of the method application is shown in Fig. 2. A circuit consisting of AND gates with the same delays (2 for the rising edge delay, and 3 for the falling edge delay), is being simulated for the maximal delay estimation. Delays are accumulated through the gates as shown in Fig. 2, by using additional signal attributes.

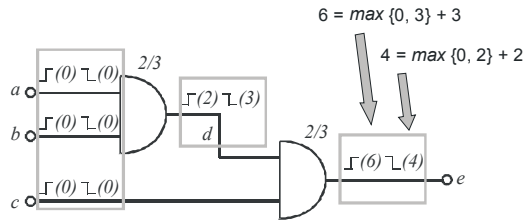

Fig. 2. A method for the maximal delay estimation.

Sequential elements such as flip-flops are modeled in a similar manner. For a network of sequential elements, it is inapplicable to calculate delays in this way if those flip-flops have clocking. The delay of the signal occurring at the output of such a circuit is estimated only according to the clock impulse. That is the reason why this method is only adequate to apply to embedded sequential circuit. The example of one is a particular flip-flop surrounded with a combinational network that can be seen in counters. Fig. 3 presents a D-flip-flop in a BCD counter, dedicated for the second digit Q_2.

In this part of the circuit it is important that data impulse at the D-input of the following flip-flop is stabile before its next clock pulse arrives. It means that delays from the previous flip-flop in a chain together with the worst case gate and flip-flop delays should fulfill the timing requirements (minimal clock pulse duration and the maximal operating frequency) of the whole counter.

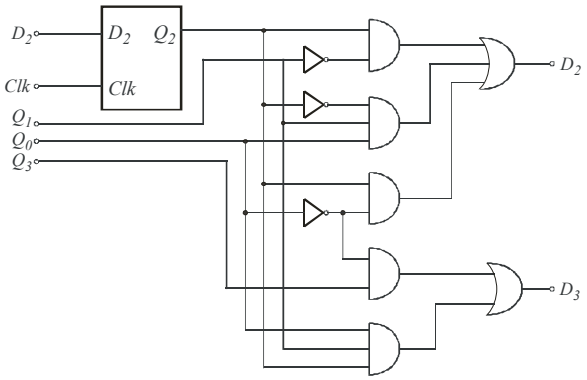

Fig. 3. Embedded sequential circuit – part of the BCD counter.

III. VHDL IMPLEMENTATION

Although VHDL-93 [9] offers the user-defined attributes, they cannot be accessed and updated from the processes during the simulation. Therefore, for the implementation of the timing simulation in the VHDL simulator, a different modeling mechanism is necessary. We propose the use of *composite signals of the record type*. Such a signal consists of logic state and timing attributes denoted with *arr*, for the information about the arrival of a particular edge and *d*, for the delay accumulation through the circuit. The record type DCSM_std_logic for scalar signals is defined in VHDL as:

```
type DCSM_std_logic is record
  statemn: std_logic;
  d0mn: time;
  d1mn: time;
  arr0mn: boolean;
  arr1mn: boolean;
```

```
    statemx: std_logic;
    d0mx time;
    d1mx: time;
    arr0mx: boolean;
    arr1mx: boolean;
end record DCSM_std_logic;
```

Assuming the input and output signals of the DCSM_std_logic type, the complete model of a two-input NAND gate can be described in VHDL as shown in Fig. 4

```
entity nandg is
   generic (tpd_hlmn : time := 0.9 ns;
      tpd_lhmn : time := 1 ns;
      tpd_hlmx : time := 0.95 ns;
         tpd_lhmx : time := 1.05 ns);
  port (out1 : out DCSM_std_logic :=
        ('0', 0.0 sec, 0.0 sec, false, false, '0', 0.0 sec, 0.0 sec,
false, false);
       in1, in2:  in DCSM_std_logic :=
        ('0', 0.0 sec, 0.0 sec, false, false, '0', 0.0 sec, 0.0 sec,
false, false));
end nandg;
architecture only of nandg is
begin
---- Logic function:
 p1: process(in1.statemn, in2.statemn)
   variable val,ex_value : std_logic := '0';
   begin
      val := in1.statemn and in2.statemn;
      val := not (val);
      if val /= ex_value then
        ex_value := val;
        case val is
          when '0' => out1.statemn <= val after tpd_hlmn;
          when '1' => out1.statemn <= val after tpd_lhmn;
          when others => end case;
      end if;
   end process p1;
   p2: process(in1.statemx, in2.statemx)
   variable val,ex_value : std_logic := '0';
   begin
      val := in1.statemx and in2.statemx;
      val := not (val);
      if val /= ex_value then
        ex_value := val;
        case val is
          when '0' => out1.statemx <= val after tpd_hlmx;
          when '1' => out1.statemx <= val after tpd_lhmx;
          when others => end case;
      end if;
   end process p2;
---- Timing simulation
   p3: process (in1.d0mn, in1.d1mn, in1.arr0mn, in1.arr1mn,
            in2.d0mn, in2.d1mn, in2.arr0mn, in2.arr1mn)
   begin
    if (in1.arr0mn or in2.arr0mn) then
    out1.d1mn   <= min(in1.d0mn, in2.d0mn) + tpd_lhmn;
    out1.arr1mn <= true;
    end if;
    if (in1.arr1mn and in2.arr1mn) then
    out1.d0mn   <= min(in1.d1mn, in2.d1mn) + tpd_hlmn;
    out1.arr0mn <= true;
    end if;
   end process p3;
  p4: process (in1.d0mx, in1.d1mx, in1.arr0mx, in1.arr1mx,
            in2.d0mx, in2.d1mx, in2.arr0mx, in2.arr1mx)
   begin
    if (in1.arr0mx or in2.arr0mx) then
    out1.d1mx   <= max(in1.d0mx, in2.d0mx) + tpd_lhmx;
    out1.arr1mx <= true;
    end if;
    if (in1.arr1mx and in2.arr1mx) then
    out1.d0mx <= max(in1.d1mx,in2.d1mx) + tpd_hlmx;
    out1.arr0mx <= true;
    end if;
   end process p4;
end only;
```

Fig. 4 VHDL implementation of the two-input NAND gate model.

Here, processes labeled *p1* and *p2* are assigned for the standard logic simulation. They model the logic function and delay function of the NAND gate. Processes labeled *p3* and *p4* are intended for the timing simulation. They monitor the timing attributes of gate inputs *inp1* and *inp2* and update the timing attributes of gate output *oupt*. Note that processes *p3* and *p4* are sensitive to the timing attributes of the input signals *inp1* and *inp2*, but not to their logic states. Similarly, processes *p1* and *p2* are sensitive to events on input signal states, but not to their timing attributes. Hence, the two types of processes are independent, logic and timing simulation can be performed separately or simultaneously.

Changes of the timing attributes of the output signal *outp* are always scheduled with zero delay in processes *p3* and *p4*. If the timing simulation is initiated at some time instant of simulation time, it will be finished at the same time instant, since the activity related to timing simulation is performed with no delay. We suggest that timing simulation should be performed in initial time instant t=0 of simulation. If no timing attribute is changed during the simulation, timing simulation is suppressed and the simulator performs standard logic simulation. Timing simulation is invoked by setting the timing attributes *arr1mn, arr1mx, arr0mn* and *arr0mx* of primary inputs to value true. Once the simulation time advances to t>0 sec, longest and shortest path delays of all signals in the circuit are available and can be logged out. One additional process in the top-level entity can be added that waits for some small time period (1 picosecond), prints worst-case path delays for rising and falling edges to the file, and then suspends to the end of the simulation.

```
---- Timing simulation:
 q.arr1mn <= true;
 q.arr0mn <= true;
 q.d0mn <= tf_ck_qmn+tsu_d_ckmn;
 q.d1mn <= tr_ck_qmn+tsu_d_ckmn;
 q.arr1mx <= true;
 q.arr0mx <= true;
 q.d0mx <= tf_ck_qmx+tsu_d_ckmx;
 q.d1mx <= tr_ck_qmx+tsu_d_ckmx;
```

Fig. 5 VHDL implementation of D-flip-flop model for timing analysis.

For timing simulation, data inputs of sequential elements are the ending points of the signal propagation paths, whereas the outputs of the sequential elements are the starting points of the signal propagation paths. That means the sequential element must assign the value true to the timing attributes *arr1mn, arr1mx, arr0mn* and *arr0mx* of output signals at time t=0 and log out the timing attributes *d1mn, d1mx, d0mn,* and *d0mx* of data inputs at some time t>0.

For example, the model of a D-type flip-flop for timing analysis can be described with the VHDL code shown in Fig. 5. The delay parameters of the flip-flop are directly added to the appropriate path delays. The propagation delays tr_ck_q and tf_ck_q are initially assigned to timing attributes *d1* and *d0* of the output *q,* for both shortest and

longest paths. The set-up times tsu_d_ckmn and tsu_d_ckmx are added to the delay of the path ending at data input *d*.

IV. Simulation Results

The efficiency of the proposed method is tested on ISCAS'85 benchmark circuits [6]. They are simulated with VHDL simulator ActiveHDL [7]. Timing simulation is performed at t=0 seconds, the results are logged at t=1 picosecond and simulation is finished at t=2 picoseconds. To be able to easily validate the results, for this experiment all gates in the ISCAS'85 circuits were first chosen to have delays *trmn*=*trmx*=*tfmn*=*tfmx*=1ns. In such a case, the shortest and the longest path delays of rising and falling edge for each signal are the same and match the topological level of that particular signal expressed in nanoseconds, as shown in column D, in Table 1. Since the gate delays are 1 nanosecond and the simulation is stopped after only 2 picoseconds, only timing simulation and initialization phase of the logic simulation is done. The results of the shortest and the longest path delays analysis are also shown in Table 1, for the simulation of the same circuits which gates have following delays: *tfmn*=0.9ns, *trmn*=1ns, *tfmx*=0.95ns, *trmx*=1.05ns. For each ISCAS'85 circuit, the simulation result was a list of *d1* and *d0,* minimal and maximal parameters of all primary output signals. The maximal of these values is given in column $D_{mx,}$ while the minimal values are given in column D_{mn} of Table 1 representing the delay of the longest and the shortest paths in the circuit.

Table 1: Results of timing simulation for ISCAS'85 benchmark circuits with a VHDL simulator.

Circuit name	*Nr. of signals*	*Nr. of inputs/ ouputs*	*Nr. of gates*	*D [ns]*	D_{fmn} *[ns]*	D_{fmx} *[ns]*	D_{rmn} *[ns]*	D_{rmx} *[ns]*
c17	11	5/2	6	3	1.9	2.95	1.9	3.05
c432	196	36/7	160	17	1.9	16.95	1.9	17.05
c499	243	41/32	202	11	0.9	11.35	1	11.45
c880	443	60/26	383	24	1.8	24	2	24.2
c1355	587	41/32	546	24	2.8	24.1	2.9	23.9
c2670	1426	233/140	1193	32	0	32.3	0	32.4
c3540	1719	50/22	1669	47	1.8	47.55	1.9	47.75
c5315	2485	178/123	2307	49	0.9	49.35	1	48.65
c6288	2448	32/32	2416	124	0.9	124	1	124
c7552	3719	207/108	3512	43	0	42.95	0	43.05

Results of the timing analysis for the circuit shown in Fig.3, are shown in Table 2. Those are worst-case delays for each circuit output. According to these data, maximal operating frequency as well as the minimal clock pulse duration can be predicted.

Table 2: Results of timing simulation for embedded sequential circuit with a VHDL simulator.

Output	D_{fmn} *[ns]*	D_{fmx} *[ns]*	D_{rmn} *[ns]*	D_{rmn} *[ns]*
D_2	1.8	4.4	2	4.6
D_3	1.8	3.35	2	3.65

Timing simulation does not significantly increase the total CPU time, as well as the memory resources.

V. Conclusion

During the design process of digital circuits it is extremely important to estimate the longest and the shortest path delays as early as possible. Design verification is one of the early phases in the design process which is usually performed using a logic simulator to verify that the circuit meets the required logic function and a timing simulator to verify that the circuit meets the required timing specifications. The method proposed in this paper combines both these tools into one by enabling the VHDL simulator to perform the timing simulation. In this way the design process is simplified and accelerated.

The problem of introducing the timing simulation into the VHDL simulator is solved by the use of composite signals of record type in VHDL. Eight additional timing attributes are associated with each signal and used only when the timing simulation is required.

Extra memory for storing the timing attributes is proportional to the total number of signals in the circuit. This storing memory is not a critical issue and the simulator performance should not be affected by the memory requirements.

The simulation results also indicate that extra simulation CPU time due to timing simulation is almost negligible for ISCAS'85 circuits. Timing simulation affects only the initialization phase of the simulation, whereas the rest of the simulation flow usually takes the most of the CPU time. The proposed method is simple for implementation in libraries of gate models used in semi-custom design methodologies.

References

[1] D. M. Maksimović and V. B. Litovski, "Logic Simulation Methods For Longest Path Delay Estimation," *IEE Proc.- Comput. Digit. Tech.*, Vol. 149, No. 2, March 2002, pp. 53-59.

[2] D. M. Maksimović and V. B. Litovski, "Tuning logic simulators for timing analysis," *Electronics Letters*, Vol. 35, No. 10, Stevenage, UK, May 1999, pp. 800-802.

[3] J. K. Ousterhout, "A Timing Analyzer for NMOS VLSI Circuits," *Proc. 3rd Caltech Conf. on VLSI*, March 1983, pp. 57-69.

[4] N. Jouppi, "TV: An NMOS Timing Analyzer," *Proc. 3rd Caltech Conf. on VLSI*, March 1983, pp. 71-85.

[5] C. Oh and M. R. Mercer "Efficient Logic-Level Timing Analysis Using Constraint-Guided Critical Path Search," *IEEE Trans. On VLSI Systems*, Vol. 4, No. 3, September 1996, pp. 346-355.

[6] *ISCAS'85 benchmark circuits*, Available: http://www.c-lab.de/~wolfgang/VHDL/models/ISCAS/

[7] *VHDL simulator*, Active-HDL, ver. 5.1, ALDEC Inc., 2003.

[8] M. Lj. Sokolović and V. B. Litovski, "Estimation of Path Delays Using VHDL Logic Simulation," presented at the *ETRAN Conference*, Budva, Serbia&Montenegro June 05–10, 2005.

[9] *VHDL Language Reference Manual 1993*. IEEE Standard 1076, 1993.