EFIKASNO ODREĐIVANJE STATISTIČKOG NAJGOREG SLUČAJA KAŠNJENJA U SLOŽENIM DIGITALNIM KOLIMA

Miljana Sokolović, Vančo Litovski, *Elektronski fakultet, Niš*

Sadržaj – *U radu je predstavljen metod za procenu najgoreg slučaja kašnjenja u digitalnim kolima, koji predstavlja ubrzanje veoma komplikovane i vremenski zahtevne Monte-Carlo analize. Ova tehnika ima mogućnost da simulira realne tehnološke procese i njihove varijacije i veoma je brza, zbog toga što se primenjuje specifičan algoritam izračunavanja vrednosti kašnjenja. Statistička analiza i procena kašnjenja implementirana je korišćenjem VHDL simulatora i Matlab programa.*

1. UVOD

Vrednosti parametara komponenata elektronskih kola podležu varijacijama iz mnogo različitih razloga. Zbog tih varijacija, odziv projektovanog kola može da postane neprihvatljiv. Ponašanje kola dobijenog u masovnoj proizvodnji često može da se razlikuje od željenog u tolikoj meri da prekorači granice prihvatljivog odziva. Kao rezultat ne može se očekivati da će odzivi svih kola da zadovoljavaju postavljene zahteve kao ni prinos od 100%, čak i u odsustvu defekata u kolu. Priroda odstupanja parametara je statistička i to u tom smislu što su dobijene vrednosti parametara slučajne u okviru intervala koji se naziva tolerancija parametra. Kao posledica toga, može se očekivati da i vrednosti odziva budu slučajne veličine koje pripadaju užem ili širem intervalu zavisno od načina preslikavanja tolerancije parametara u tolerancije odziva [1].

Interval u kome može da se nađe vrednost parametra, nezavisno od načina i uzroka promene naziva se tolerancija [1]. Ako parametri variraju, variraće i odziv. Apsolutnom tolerancijom odziva naziva se interval u kome se za date tolerancije parametara, može naći vrednost odziva. Realizacijom preslikavanja odnosno izračunavanjem tolerancija odziva, mi obavljamo analizu tolerancija.

Sa skaliranjem tehnologije, vremenska verifikacija postaje sve teži zadatak. Razlog tome su promene parametara gejtova i veza. Zbog toga statistička vremenska analiza postaje neizostavni metod. Napredni alati za vremensku analizu moraju biti sposobni da uračunaju i promene u kašnjenjima kola koje su posledica mnogo različitih izvora varijacija. Ovi izvori se mogu klasifikovati u varijacije usled proizvodnog procesa, faktore okruženja, temperaturski uticaj i fenomene unutar komponenata kao što su npr. elektro migracija i slično. Statička vremenska analiza predstavlja analizu najgoreg slučaja. Kako se broj izvora varijacija povećava, broj potrebnih statičkih analiza najgoreg slučaja eksponencijalno raste. Kako je nemoguće analizirati najgore slučajeve u svim ravnima parametara, neki od neanaliziranih slučajeva mogu da rezultuju defektima, nakon što se čip proizvede. U radu [2] pored uticaja varijacija tehnološkog procesa na kašnjenje veza, analiziran je i uticaj različitih izvora varijacija na performanse gejtova. U ovom radu se svaki pojedinačni kvant vremena posmatra kao funkcija globalnih izvora varijacje. Ovako se dobija mnogo precizniji model kašnjenja gejtova. Ovaj pristup je između 100 i 200 puta brži od *Monte-Carlo* analize, a greška je smanjena i do 20% u pojedinim slučajevima. Međutim, postupak takvog izračunavanja, i dalje traje veoma dugo. Potrebno je izračunati osetljivosti svih parametara jednog gejta na sve moguće izvore varijacija. U postupku koji se predlaže u ovom radu vremenska analiza traje neuporedivo brže, jer se izborom odgovarajuće devijacije i srednje vrednosti kašnjenja gejtova, sva ta izračunavanja mogu izbeći.

Postojeći alati za vremensku analizu rade se kombinacionim kolima koja se sastoje od primitivnih gejtova. Međutim, veliki digitalni sistemi često se sastoje i od kompleksnih gejtova. U radu [3], istraživani su načini za vremensku analizu ovakvih kola. Prvi pristup ovom problemu je da se ustanovi kriterijum za senzitovanje puta kod kola koja su sastavljena od kompleksnih gejtova. Drugi pristup je da se složeni gejtovi prvo transformišu u više prostih, a zatim da se korišćenjem postojećih alata obavi standardna vremenska analiza.

Industrijski sistemi za vremensku analizu danas se oslanjaju na statičku vremensku verifikaciju, zbog toga što komercijalni alati podržavaju samo takve opcije. Međutim, pokušaji da se optimizuje kolo na osnovu strukturnog određivanja najdužeg puta, rezultira povećanjem površine i potrošnje čipa. Drugi problem koji proizilazi iz statičke vremenske verifikacije na nivou modula je što se stvarni najduži putevi na ovom nivou pokazuju najčešće lažnim kada se pređe na viši nivo. Vremenska verifikacija celog integrisanog kola mora se obaviti na funkcionalnom nivou, zbog toga što je potrebno odrediti radnu frekvenciju čipa u toku projektovanja. Optimizacija nakon proizvodnje nije moguća. U radu [4] razvijeno je efikasno rešenje za vremensku verifikaciju i generisanje testa za kašnjenje na nivou čipa, na taj način što je razmatrana hijerarhijska dekompozicija digitalnog sistema. Ovaj metod zahteva komercijalne alate za generisanje i verifikaciju testa. Hijerarhijski pristup generisanju testa za ispitivanje kašnjenja abstrahuje ostatak velikog čipa u čisto logičko ponašanje ugrađenog modula. Dobijeno smanjenje kompleksnosti dozvoljava da se identifikuju kritični putevi na nivou čipa i da se nađe test za ispitivanje kašnjenja koji će senzitovati te kritične puteve. Nedostatak ovakvog rešenja jeste namogućnost uključivanja tolerancija parametara komponenata. Sa druge strane on je zasnovan na primeni poznatog skupa pobudnih signala, kako na nivou modula, tako i na nivou celog čipa.

U ovom radu će najpre biti navedeni poznati metodi analize tolerancija. Zatim će biti opisan novi metod procene tolerancija kašnjenja koji objedinjava nekoliko postojećih. Da bi bilo moguće primeniti novi metod, potrebno je usvojiti novi model svakog logičkog gejta. Modeliranje gejtova je takođe izloženo u radu. Sa takvim modelom moglo se pristupiti implementaciji metoda koji se zasniva na primeni mnogostrukih simulacija, i analizi dobijenih rezultata. VHDL implementacija metoda će biti data u narednom odeljku. Na kraju će biti prikazani rezultati nove analize tolerancija koji su dobijeni posebnim Matlab programima pri čemu su kao kola za verifikaciju metoda poslužila pojedina karakteristična ISCA-S89 benchmark kola [5].

2. METOD PROCENE

Metodi za analizu tolerancija mogu se grupisati u dve kategorije [6]: analize najgoreg slučaja i analize koje ne traže najgori slučaj. U prvom slučaju čine se napori da se pronađe kolo koje će imati najverovatnije najgori odziv u odnosu na nominalni sa stanovišta preslikavanja tolerancija parametara.

Dobijeni rezultat, mada ne može da se koristi i za sintezu tolerancija, govori o osobinama projekta i može da bude direktno poređen sa prihvatljivim odzivom koji je zadat. Druga kategorija metoda za analizu tolerancija obuhvata metode uzorkovanja i direktne metode. Ako se koriste direktne metode to znači da postoje formule za preslikavanje tolerancija parametara u tolerancije odziva. Upotreba ovih metoda je obično ograničena na slučajeve kada su priraštaji parametara mali. Među metodama uzorkovanja prepoznaje se najmoćniji metod statističkog uzorkovanja poznat pod imenom *Monte-Carlo*. On teži da simulira prirodni fenomen uzorkovanja koji se dešava tokom montaže elektronskih uređaja i otuda potiče njegova moć i njegova popularnost [1].

Metod koji se ovde predlaže objedinjuje metod najgoreg slučaja i *Monte-Carlo* metod u proceni tolerancija odziva. Metod se odnosi na kašnjenja u složenim digitalnim kolima, ali se može primeniti i za procenu tolerancija nekih drugih performansi kao što je na primer potrošnja snage.

Pre početka simulacije, potrebno je za svaku vrstu gejtova usvojiti statistički model najgoreg slučaja kašnjenja. O modelovanju gejtova će kasnije biti više reči. Zatim se VHDL simulacijom procenjuje minimalno i maksimalno (worst case) kašnjenje rastuće i opadajuće ivice do svih izlaza u kolu prema metodi datoj u [7], [8]. Kako pri ovim simulacijama nema potrebe uzimati u obzir logičku funkciju gejtova, već samo njihova kašnjenja, tako one traju veoma brzo. Treba napomenuti da za ovakve simulacije nije potrebno zadavati nikakvu pobudu, i u obzir se uzimaju sve moguće tranzicije signala kao i kombinacije najgorih slučajeva kašnjenja. Svaki gejt slučajno generiše svoje maksimalno i minimalno kašnjenje prednje i zadnje ivice po *Gauss*-ovoj raspodeli, sa definisanom srednjom vrednošću i devijacijom, prema izrazu (1),

$$\phi(p) = \{\mathbf{exp}[-(p-\mu_p)^2/(2\sigma_p^2)]\}/[\sigma_p\sqrt{2\pi}] \qquad (1)$$

gde je μ_p srednja vrednost ili matematičko očekivanje, a σ^2 varijansa slučajne promenljive p [1].

Radi se onoliko simulacija kola istovremeno koliko je dovoljno da bi se koretno obavila statistička anlaliza rezulrata (600). Zatim se za tako dobijene vrednosti kašnjenja obavlja usrednjavanje, prema izrazu (2) i određuje devijacija najgoreg slučaja kašnjenja prednje i zadnje ivice signala na svim izlazima kola, prema izrazu i (3) [1].

$$\mu_p = [\sum_{i=1}^{N} p_i]/N \qquad (2)$$

$$\sigma^2 = [\sum_{i=1}^{N} (p_i - \mu_p)^2]/N \qquad (3)$$

Svaki gejt modeluje se sledećim parametrima: minimalno kašnjenje postavljanja rastuće ivice signala na izlazu kola – *trmn*, maksimalno kašnjenje postavljanja rastuće ivice signala na izlazu kola – *trmx*, minimalno kašnjenje postavljanja opadajuće ivice signala na izlazu kola – *tfmn*, maksimalno kašnjenje postavljanja opadajuće ivice signala na izlazu kola – *tfmx*. Pošto se kašnjenje pojedinačnog gejta slučajno bira, onda gore navedene vrednosti predstavljaju srednje vrednosti u toj raspodeli. Kao zajednički činilac u raspodelama svih navedenih parametara, javlja se i standardna devijacija σ, koja je u našem slučaju izabrana da bude 3%. Naravno, sve ove vrednosti moguće je podešavati. Ova tolerancija parametara posledica je tolerancija parametara procesa izrade integrisanih kola, kao što su npr. debljina oksida i koncentracija nosilaca u podlozi. Treba još reći da su gejtovi oslobođeni svih izračunavanja koja se tiču npr. njihove logičke funkcije. Proces kojim se opisuje dodeljivanje maksimalnih i kašnjenja opadajuće i rastuće ivice jednog dvoulaznog NI kola prikazan je na slici 1.

Ulazi gejta označeni su sa *a* i *b*, dok je izlaz gejta označen sa *f*. Maksimalna propagaciona kašnjenja za rastuću i opadajuću ivicu izlaza *f* označena su sa *trmx* i *tfmx* i ona se slučajno generišu po *Gauss*-ovoj raspodeli. Svaka opadajuća tranzicija signala na bilo kom ulazu kola rezultira rastućom tranzicijom izlaznog signala gejta. S druge strane, rastuća tranzicija na nekom ulazu može da izazove opadajuću tranziciju signala na izlazu gejta, samo ako je prethodo na drugi ulaz gejta takođe pristigla rastuća tranzicija signala [7], [9]. U ovakav model kašnjenja mogu se ugraditi i efekti različitih nagiba signala, kapacitivnosti opterećenjai drugi.

```
process (a, b) {
    // update falling edge delay of signal f:
    if (arr0mx(a) .OR. arr0mx(b))
      d1mx(f) := MAX(d0(a),d0(b)) + rand (tfmx);
      arr1mx(f) := 'true';
      f <= an_event;    }
    // update rising edge delay of signal f:
    if (arr1mx(a) .AND. arr1mx(b)) {
      d0mx(f) := MAX(d1(a),d1(b)) + rand (trmx);
      arr0mx(f) := 'true';
      f <= an_event;    }  }
```

Sl.1. *Model dvoulaznog NI kola za procenu maksimalnog kašnjenja prednje i zadnje ivice signala*

3. VHDL (I MATLAB) IMPLEMENTACIJA METODA

Prilikom VHDL implementacije modela gejtova najpre je bilo potrebno realizovati generator slučajnih brojeva po normalnoj raspodeli. Standardni VHDL simulator ima dostupne procedure za generisanje slučajnih brojeva sa uniformnom raspodelom na intervalu [0,1]. Da bi se najpre na osnovu takve raspodele dobili brojevi sa normalnom raspodelom, trebalo je uvesti odgovarajuće procedure u proces generisanja. Ovaj postupak sastoji se od sledećih koraka [10]:

- izabrati dva broja u_1 i u_2 sa uniformnom raspodelom na segmentu [0,1]
- izvršiti preslikavanje opsega [0,1] u opseg [-1,1] uvođenjem smena $v_1=2u_1-1$ i $v_2=2u_2-1$
- izračunati R prema jednačini (4)

$$R = v_1^2 + v_2^2 \qquad (4)$$

- od ovako nastalih parova odabrati one koji zadovoljavaju $R<1$
- na izabrane parove v_1, v_2 primeniti sledeće smene

$$X_S = \sqrt{\frac{-2\ln(R)}{R}} \cdot v_1 \text{ i } X_P = \sqrt{\frac{-2\ln(R)}{R}} \cdot v_2 \qquad (5)$$

```
function gauss_rng return real is
        variable u1, u2, v1, v2, r, q, p: real;
    begin
          loop     u1:=rand;
                   u2:=rand;
                   v1:=u1*2.0 - 1.0;
                   v2:=u2*2.0 - 1.0;
                   r:=v1*v1 + v2*v2;
                   exit when r<1.0;
          end loop;
          q:=log2(r);
          p:= (sqrt( (0.0-2.0)*q/r ))* v1 ;
          return p;
    end function gauss_rng;
```

Sl.2. *VHDL implementacija generatora slučajnih brojeva po normalnoj raspodeli*

VHDL implementacija ovakvog generatora slučajnih brojeva prikazana je na slici 2. Funkcija *rand* koja se javlja u okviru ove procedure služi za generisanje slučajnih brojeva na intervalu [0,1], sa normalnom raspodelom. Detaljniji opis ove funkcije može se naći u [11].

Funkcija generisanja slučajnih brojeva po *Gauss*-ovoj raspodeli poziva se po 4 puta u modelu svakog gejta. Tom

prilikom, svakom gejtu posebno, dodeljuju se 4 nove promenljive koje se tiču srednjih vrednosti raspodele μ_p i devijacije parametara σ. Tako se uvođenjem odgovarajućih smena dobija *Gauss*-ova raspodela, data jednačinom (6):

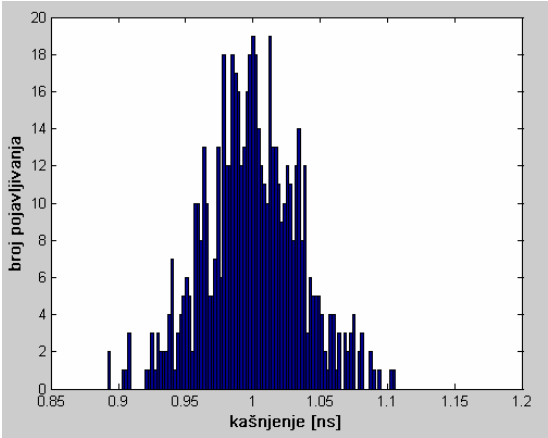
$$Y = \mu_p + \sigma X \,. \tag{6}$$

Proces vremenske analize implementiran u okviru jednog NI kola kojim se dodeljuju i određuju maskimalna kašnjenja rasuće i opadajuće ivice signala na izlazu gejta prikazan je na slici 3. Sam gejt sadrži još jedan sličan proces kojim se dodeljuju i procenjuju minimalna kašnjenja rastuće i opadajuće ivice. Svaki od ovakvih opisa logičkih kola uključen je u odgovarajuću biblioteku koja se poziva u opisu netliste složenog digitalnog sistema na čijim se izlazima procenjuju kašnjenja.

```
p2: process (in1.d0mx, in1.arr0mx,in1.d1mx,
    in1.arr1mx,    in2.d0mx,    in2.arr0mx,    in2.d1mx,
    in2.arr1mx)
    variable r,p: real;
    begin
    r:= (1.05 + (0.03*(gauss_rng)));
         p:= (0.95 + (0.03*(gauss_rng)));
    if (in1.arr0mx or in2.arr0mx) then
         out1.d1mx <= max(in1.d0mx, in2.d0mx)+r;
         out1.arr1mx <= true;
    end if;
    if (in1.arr1mx and in2.arr1mx) then
         out1.d0mx <= max(in1.d1mx,in2.d1mx)+p;
         out1.arr0mx <= true;
    end if;
end process p2;
```

Sl.3. *VHDL model NI logičkog kola*

Da bi se verfikovala efikasnost primenjenog modela, načinjeno je probno kolo koje se sastoji samo od jednog ovako opisanog NI kola. Ovo kolo simulirano je 600 puta i rezultati slučajno generisanog kašnjenja prednje ili zadnje ivice izlaznog signala prikazani su histogramski na Sl. 4. U ovom slučaju srednja vrednost u raspodeli iznosi 1 ns, a devijacja 3 %. Na apscisi su prikazana kašnjenja u [ns], a na ordinati broj odziva iz zadatog opsega.



Sl.4. *Histogram odziva NI logičkog kola*

Pri implementaciji metoda neophodno je u simulaciju uvesti takve signale koji će nositi nekoliko informacija, a ne kao u standardnim simulacijama samo logičku funkciju. U našem slučaju, za implementaciju metoda iskorišćeni su kompozitni tipovi signala. Svaki takav signal sadrži nekoliko tipova informacija i to tipa real i tipa boolean. Podaci tipa real u okviru jednog signala, nose informaciju o do tad nakupljenom kašnjenju u kolu. Podaci tipa boolean, nose informaciju o tome da li je do posmatranog gejta stigla odgovarajuća tranzicija. Ukoliko je informacija o pristizanju pozitivna, u gejtu se izračunava rezultujuće kašnjenje (ovo kašnjenje zavisi od onih koja su pristigla do ulaza gejta i od novogenerisanog kašnjenja samog gejta).

Netlista kola koje se analizira potpuno je ista kao i za standardnu logičku simulaciju. Za analizu tolerancija kola sasvim je dovoljno poznavti koji su gejtovi prisutni u složenom digitalnom kolu i njihove međusobne veze. Sa takvom netlistom kola, i sa već opisanim modelima svih gejtova može se pristupiti simulacijama i analizi.

```
type   input_mat   is   array   (1   to   600)   of
SDA_std_logic_vector(0 to 59);

type   output_mat   is   array   (1   to   600)   of
SDA_std_logic_vector(0 to 25);

signal inputs: input_mat;

signal outputs: output_mat;

signal  J: integer;
begin
   G2:for J in 1 to 600 generate
   c880_inst: c880 port map (inp => inputs(J), outp =>
outputs(J));
   inputs(J) <= (others => (0.0, 0.0, true, true, 0.0,
0.0, true, true));
   end generate;
log_timing1: process
   use std.textio.all;
   file    log:    text    open    write_mode    is
   "c880mn_r.statdel";
        variable line_1: line;
        variable I, J:integer;
        variable  izlaz:  SDA_std_logic_vector(0  to
        25);
variable delay_mn_r : real;
   begin
   wait for 1 ps;
        for J in 1 to 600 loop
           for I in 0 to 25 loop
           izlaz := outputs(J);
           delay_mn_r:= izlaz(I).d1mn;
write (line_1,delay_mn_r,left, 15);
end loop;
        writeline (log, line_1);
        end loop;
        wait;
   end process log_timing1;
```

Sl.5. *Inicijalizacija početnih stanja signala i proces vremenske analize testbench kola za c880 benchmark kolo*

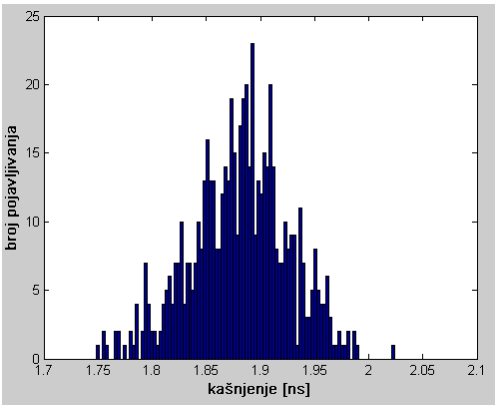
Da bi se kolo simuliralo i analiziralo, potrebno je u VHDL-u napisati poseban *testbench* program. U tom programu se netlista složenog digitalnog kola instancira nekoliko stotina puta. Sada se za svaki odgovarajući pojedinačni ulaz svih tih kola prave matrice za iniciranje njihovog početnog stanja i same simulacije. Svi odzivi logičke analize takođe se pamte u vidu matrice. Matrice ulaznih i izlaznih signala definisane su novim tipom promenljive *input_mat* i *output_mat*. Ovo je sve prikazano u kodu na Sl. 5. Ovaj kod sadrži i opis procesa kojim se obavlja vremenska analiza – ***log_timing1***, a koji se koristi za određivanje minimalnog kašnjenja rastuće ivice izlaznih signala ISCAS89 benchmark kola c880. Ovo kolo sadrži 60 ulaza i 26 izlaza. Svi rezultati ove analize upisuju se u tekstualni fajl pod imenom *c880mn_r.statdel*.

Slični procesi predviđeni su i za simulacije za određivanje statističkog maksimalnog kašnjenja rastuće ivice i statističkih maksimalnih i minimalnih kašnjenja opadajuće ivice svih izlaza ovog kola.

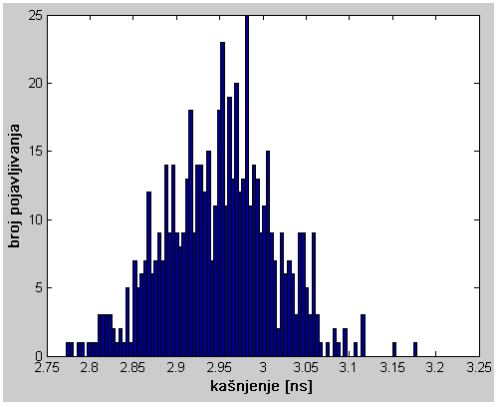
4. REZULTATI

Kada se svako složeno digitalno kolo simulira 600 puta, očekuje se ogromna količina podataka. Sve te podatke potrebno je na odgovarajući način predstaviti. Kod kola koja imaju mali broj izlaza, dobijeni statistički podaci mogu se prikazati u vidu histograma. Na Sl. 6 prikazani su histogrami odziva ISCAS c17 benchmark kola. Ovo kolo ima 2 izlaza, i za svaki od tih izlaza moguće je generisati histograme maksimalnog i minimalnog kašnjenja prednje i zadnje ivice. Na ovoj slici prikazani su samo histogrami minimalnog kašnjenja rastuće ivice i maksimalnog kašnjenja opadajuće ivice za jedan od izlaza ovog kola.

Međutim, kada se analiziraju kola koja imaju mnogo više izlaza, kao što je ISCAS c880 kolo, koje ima 26 izlaza, histogramski prikaz rezultata nije dobro rešenje. U ovom slučaju, napisan je još jedan Matlab program koji na osnovu svih sakupljenih rezultata određuje devijaciju kašnjenja pojedinih izlaza. Rezultati ovog izračunavanja prikazani su u tabeli 1. Pored vrednosti za devijaciju minimalnog kašnjenja opadajuće, *mn_f*, minimalnog kašnjenja rastuće, *mn_r*, maksimalnog kašnjenja opadajuće, *mx_f* i maksimalnog kašnjenja rastuće, *mx_r*, u tabeli su predstavljeni i topološki nivoi.



a)



b)

Sl.6. *Histogrami pojedinih odziva ISCAS 89 kola c17: a) mn_r1, b) mx_f1*

Tabela.1. *Standardna devijacije kašnjenja ISCAS c880 benchmark kola*

izlaz	t.l.mn	mn_f	mn_r	t.l.mx	mx_f	mx_r
1	2	0.048589	0.052759	2	0.050674	0.050610
2	2	0.050828	0.050476	2	0.049100	0.050790
3	2	0.051576	0.049895	2	0.049352	0.048982
4	2	0.052952	0.049668	2	0.049607	0.051089
5	3	0.065288	0.060681	3	0.061082	0.062817
6	3	0.059224	0.057522	3	0.058721	0.056839
7	3	0.058623	0.062774	3	0.058978	0.061083
8	3	0.060815	0.058665	3	0.061629	0.062541
9	3	0.062117	0.058670	3	0.062595	0.063128
10	2	0.049499	0.049472	3	0.064401	0.060103
11	3	0.063138	0.061094	4	0.070156	0.072417
12	4	0.071049	0.076989	4	0.073077	0.073517
13	4	0.070504	0.068897	4	0.071786	0.069362
14	4	0.067937	0.069778	4	0.069788	0.068860
15	3	0.063707	0.059456	4	0.074612	0.072783
16	5	0.071932	0.073721	11	0.091900	0.089868
17	5	0.073976	0.073108	11	0.093782	0.096104
18	5	0.082620	0.080388	18	0.128925	0.132698
19	6	0.080513	0.080122	20	0.122495	0.126766
20	5	0.075704	0.080168	20	0.124431	0.127338
21	5	0.078213	0.080897	20	0.132566	0.136552
22	5	0.079373	0.080192	20	0.139965	0.138840
23	6	0.082751	0.084195	22	0.129968	0.136279
24	6	0.080811	0.077761	24	0.148336	0.137821
25	6	0.080049	0.074921	24	0.135780	0.139389
26	6	0.078273	0.080604	24	0.136592	0.140061

puteva kroz koje prolaze signali sa ovakvim devijacijama kašnjenja, *t.l.mn* i *t.l.mx*. Ove vrednosti dobijene su primenom metoda datog u [7]. Na osnovu ovih rezultata može se sagledati uticaj dužine puta na konačnu devijaciju kašnjenja nekog izlaznog signala.

5. ZAKLJUČAK

U radu je predložen novi metod analize tolerancija kašnjenja kod složenih digitalnih kola. Metod objedinjuje već postojeće metode analize najgoreg slučaja i *Monte-Carlo* metod, tako da se kao rezultat dobijaju statistički najgori slučajevi kašnjenja. Metod je implementiran u VHDL kôdu, i verifikovan na pojedinim ISCAS89 benchmark kolima.

LITERATURA

[1] V. B. Litovski, and M. Zwolinski, *VLSI circuit simulation and optimization,* Chapman and Hall, 1997.

[2] S. Abbaspour, H. Fatemi, and M. Pedram, "VGTA: Variation – Aware Gate Timing Analysis", *Proc. of the IEEE International Conf. on Computer Design 2005*, San Jose, California.

[3] Y. C. Hsu, H. C. Chen, S. Sun, and D. Du: "Timing Analysis of Combinational Circuits Containing Complex Gates" *Proc. of the IEEE International Conf. on Computer Design 1998*, San Jose, California.

[4] A. Krishnamachary, A. Abraham, and R. S. Tupuri: "Timing verification and Delay Test Generation for Hierarchical Designs", *Proc. of the 14th International Conference on VLSI Design*, 2001.

[5] http://courses.ece.uiuc.edu/ece543/iscas89.html

[6] R. Spence and R. Soin, *Tolerance design of Electronic circuits*, Wokingham, England: Addison-Wesley Publ. Comp, 1988.

[7] M. Sokolović and D. Maksimović, "Estimation of Path Delay Using VHDL Logic Simulator," in *Proc. of the XLIX Conf. of ETRAN,* Budva, 2005, vol. 1. pp 99 – 102.

[8] M. Sokolivić and V. Litovski, "Using VHDL Simulator to Estimate Logic Path Delays in Combinational and Embedded Sequential Circuits," *Proc. of the IEEE Region 8 EUROCON 2005 Converence,* pp. 547-550.

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

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

[11] M. Zwolinski, *Digital System Design with VHDL,* Pearson: Prentice Hall, 2004.

Abstract – A method for worst case delay path estimation in digital circuit that implements computationally effective Monte-Carlo analysis is presented in this paper. This technique has the ability to simulate the real fabricating condition's variations of a particular technology and is at the same time very fast, since efective delay estimation algorithm is embedded within the Monte-Carlo loop. The method is implemented using VHDL simulator and a Matlab program.

EFFICIENT COMPUTATION OF THE STATISTICAL WORST CASE DELAY IN COMPLEX DIGITAL CIRCUITS

Miljana Sokolović, Vančo Litovski