Elektrotehniški vestnik 74(1-2): 49-54, 2007
Electrotechnical Review: Ljubljana, Slovenija

Neponovljivo obnašanje programov na različnih procesorjih

Matej Šalamon, Tomaž Dogša

Univerza v Mariboru, Fakulteta za elektrotehniko računalništvo in informatiko, Smetanova 17, 2000 Maribor
E-pošta: matej.salamon@uni-mb.si

Povzetek. Članek obravnava problem neponovljivega obnašanja programov, inštaliranih na računalnikih z različnimi procesorji. Predlagan je kriterij za ponovljivost in način detektiranja neponovljivosti s posebnim testnim programom. Ta vsebuje testne primere, s katerimi je mogoče napovedati, ali lahko pričakujemo na računalnikih z različnimi procesorji neponovljivo obnašanje poljubne programske opreme. Podrobneje so predstavljeni tudi morebitni vzroki za neponovljivost.

Ključne besede: neponovljivost, procesorji, testiranje, simulator električnih vezij, Chujev oscilator

Software non-repeatability behaviour in different processors

Extended abstract. When we run the same program on computers with the same or different processors, we expect its behaviour would be the same. When this is not the case, the program non-repeatability occurs.
Non-repeatability can remain hidden to users, if differences between results of different processors are very small. If differences increase in time, execution code behaviour in different processors will no longer be the same.
This paper deals with the software non-repeatability problem and its detection with a test program. In section 2 we propose a software repeatability criterion and present an example of circuit simulator non-repeatability. The test program for the software non-repeatability prediction, installed on a computer with different processors, is described in section 3. The test program contains different test cases which verify the processors compliance with IEEE 754 and IEEE 854 standards. Results in section 3 show that the reasons for software non-repeatability are errors in the program, compiler errors and execution differences of the same mathematical operations in different processors. The latter occur when the program is running on a computer with a processor whose characteristics do not comply with the above standards.
Namely, processor manufacturers build additional functions in their processors in order to achieve better performance and attractiveness for their market, although their functionality is not always in accordance with the IEEE standards.

Prejet 12. september, 2006
Odobren 17. januar, 2007

Key words: non-repeatability, processors, testing, circuits simulators, Chua's oscillator.

1. Uvod

Izdelovalci različne programske opreme zelo pogosto predpisujejo strojno opremo, ki bo zagotavljala dovolj hitro in pravilno delovanje programa. Pri tovrstni usklajenosti programske in strojne opreme uporabnik pričakuje tudi ponovljivo obnašanje programske opreme na računalnikih z različnimi, predpisanimi procesorji. Ponovljivo obnašanje programa torej pomeni, da program, inštaliran na računalnikih z enakimi ali različnimi procesorji, daje zmeraj popolnoma enake rezultate. V nasprotnem primeru je obnašanje programa neponovljivo.

Temeljno načelo znanosti je ponovljivost poskusov in opazovanj. Ponovljivost je lastnost poskusa oziroma opazovanja, da dá v kar se da podobnih razmerah enake oziroma vsaj približno enake rezultate, kar pomeni, da je variiranje med poskusi majhno oziroma zanemarljivo. Znanstvene dokaze je mogoče podati le, če je zagotovljena ponovljivost poskusov in opazovanj.

Podobno kot velja za rezultate poskusov, mora veljati tudi za obnašanje programske opreme. Če je na dveh računalnikih z različnima procesorjema inštalirana enaka programska oprema, mora biti njeno obnašanje oziroma odziv na obeh računalnikih enak, sicer je obnašanje programske opreme neponovljivo.

Neponovljivost je lahko izražena na različne načine. Lahko je prikrita, če so odstopanja med obnašanjem programa, inštaliranega na računalnikih z različnimi procesorji, vedno tako majhna, da jih uporabnik sploh ne opazi. V tem primeru je njena detekcija precej

zahtevna, saj zahteva natančno primerjavo procesiranih podatkov v procesorjih.

Neponovljivost postane očitna, če se odstopanja med primerljivimi podatki procesorjev s časom večajo. V takšnem primeru postane čez čas obnašanje enake izvršne kode programa, na različnih procesorjih, zelo različno. Zaradi velikih odstopanj je neponovljivost zelo opazna oziroma jo je mogoče enostavno zaznati. Nastopi lahko tudi situacija, ko program na enem računalniku popolnoma odpove – »se obesi«, medtem ko se na drugem obnaša popolnoma pričakovano.

V članku predstavljamo problem neponovljivega obnašanja programske opreme in način detekcije neponovljivosti s posebnim testnim programom. Drugi del članka je namenjen opisu predlaganega kriterija za ponovljivo obnašanje programske opreme in primer neponovljivega obnašanje programska - simulatorja električnih vezij. V tretjem delu je opisan testni program, ki omogoča napoved morebitnega neponovljivega obnašanja poljubne programske opreme na računalnikih z različnimi procesorji. Na koncu so podani tudi rezultati izvedenega testiranja in možni vzroki za neponovljivo obnašanje programske opreme.

2. Kriterij za ponovljivo obnašanje programske opreme

Detektiranje neponovljivosti zahteva zajemanje podatkov, procesiranih v primerjanih procesorjih in njihovo primerjavo.

Zajemanje podatkov je razmeroma preprosto, če že sam program omogoča zapisovanje rezultatov (številčnih vrednosti) v izhodno datoteko ali, če razpolagamo z njegovo izvorno kodo. Slednjo je potrebno modificirati tako, da omogoča prestrezanje in shranjevanje številčnih vrednosti posameznih spremenljivk.

Kadar ne obstaja nobena od teh dveh možnosti, je zajemanje podatkov precej bolj zahtevno. V takšnih primerih, ki pa so zelo pogosti, je potrebno podatke spremljati s pomočjo razhroščevalnika na nivoju strojnega jezika.

Za detekcijo neponovljivega obnašanja programa P, inštaliranega na računalniku s procesorjem A in računalniku s procesorjem B predlagamo naslednje: podatke oziroma vrednosti spremenljivk, uporabljene pri procesiranju s procesorjem A označimo z $a_1, a_2, \ldots a_n$ in jih zapišimo v obliki vektorja **a**:

$$\mathbf{a} = [a_1, a_2, \ldots a_n], \tag{1}$$

vrednosti, s katerimi procesira procesor B, pa zapišemo v obliki vektorja **b**:

$$\mathbf{b} = [b_1, b_2, \ldots b_n]. \tag{2}$$

Odstopanja med vrednostmi v procesorju A in B ovrednotimo s pomočjo razdalje med vektorjema **a** in **b**:

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} |a_i - b_i| . \tag{3}$$

Kriterij s pomočjo katerega sklepamo, ali je obnašanje programa P na računalniku s procesorjem A in B *ponovljivo*, je razdalja med primerjanima vektorjema **a** in **b**. Če je ta enaka nič:

$$d(\mathbf{a}, \mathbf{b}) = 0 . \tag{4}$$

je obnašanje programa P ponovljivo, sicer je njegovo obnašanje *neponovljivo*. Obnašanje programa P bo torej ponovljivo, če bodo vse številčne vrednosti, ki nastopajo pri izvajanju računskih operacij v procesorju A popolnoma enake tistim v procesorju B.

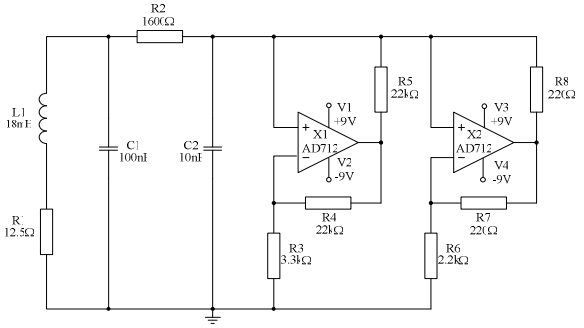
Oglejmo si praktični primer neponovljivega obnašanja programske opreme.

2.1 Primer neponovljivega obnašanja simulatorja električnih vezij

Simulator električnih vezij SPICE je eno izmed nepogrešljivih orodij načrtovalcev integriranih vezij. Gre za kompleksen program, od katerega se pričakuje ponovljive rezultate, če ga inštaliramo na računalnikih z enakimi ali različnimi in zanj predpisanimi procesorji.

Z raziskavami smo ugotovili, da temu zmeraj ni tako. V okviru tega prispevka si oglejmo nekatere izsledke izvedenih raziskav, podrobneje opisanih v literaturi [7] in [8].

S simulatorjem SPICE[1] smo simulirali kaotični Chujev oscilator (slika 1) in ugotovili, da se rezultati časovne analize tega vezja, na računalnikih z različnimi procesorji, nekaj časa ujemajo, nakar pričnejo vse bolj odstopati.

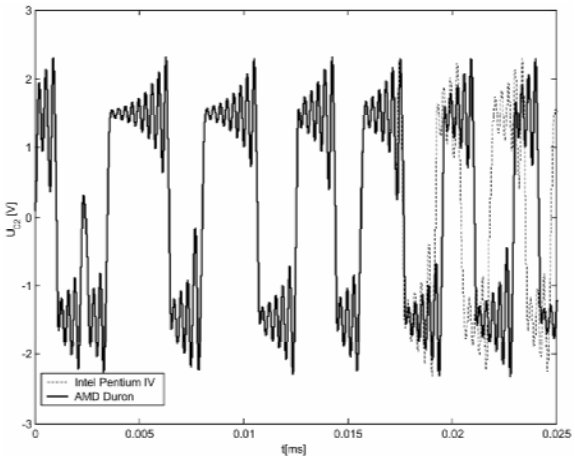


Slika 1: Chujev oscilator [6] – testno vezje

Figure 1: Chua's oscillator [6] – testing circuit

Primer tovrstnega odstopanja je prikazan na sliki 2. Časovna poteka napetosti na kondenzatorju C2 s časom divergirata, kar kaže na neponovljivo obnašanje testnega vezja, simuliranega z enakim simulatorjem na računalnikih z dvema različnima procesorjema Intel Pentium IV in AMD Duron.

[1] *ICAP/4* verzija 7.51 in 8.3.10.



Slika 2: Časovna poteka napetosti na kondenzatorju C2 v Chujevem oscilatorju, simuliranem na računalnikih s procesorjem AMD Duron in Intel Pentium IV

Figure 2: Voltage time-domain waveforms on capacitor C2 in the Chua's oscillator, simulated on computers with AMD Duron and Intel Pentium IV processors

Ugotovili smo, da je neponovljivost zaznavna le, če je obnašanje vezja kaotično in če je izbran dovolj velik čas trajanja analize prehodnega pojava [7]. Če je osciliranje Chujevega oscilatorja periodično ali če je kljub kaotičnemu režimu delovanja izbran prekratek čas trajanja analize prehodnega pojava, dejanske neponovljivosti simulacij ni mogoče zaznati. V takšnih situacijah lahko ostane dejanska neponovljivost prikrita.

3. Testiranje neponovljivosti

V zgornjem primeru smo razpolagali s programom, ki je omogočal shranjevanje izračunanih vrednosti v izhodno datoteko in s tem preprosto primerjavo rezultatov simulacij.

Ker večina programov nima te možnosti, je zajemanje oziroma shranjevanje podatkov precejšnja težava. Zato smo izdelali samostojen program, s katerim je mogoče napovedati, ali lahko na računalnikih z različnimi, izbranimi procesorji pričakujemo neponovljivo obnašanje poljubne programske opreme.

Testni program vsebuje tri testne primere in omogoča shranjevanje njihovih vmesnih in končnih rezultatov v izhodno datoteko. Omogoča tudi primerjavo datotek oziroma detekcijo neponovljivosti na podlagi predlaganega kriterija po enačbi (4).

Ker je izbor testnih primerov ključnega pomena pri detekciji neponovljivosti, si najprej oglejmo njegove podrobnosti.

3.1 Izbor testnih primerov

Testne primere za odkrivanje neponovljivosti smo izbrali na podlagi standarda IEEE 754 in IEEE 854, katerih namen je poenotiti računske operacije in lastnosti procesorjev različnih izdelovalcev. Standarda obravnavata računanje s števili s plavajočo vejico: IEEE 754 obravnava binarno aritmetiko[2], IEEE 854 pa decimalno aritmetiko[3]. Upoštevajo ju skoraj vsi današnji izdelovalci procesorjev [1], [2]. Standarda natančno določata [3], [4]:

- format zapisa števil s plavajočo vejico: format z enojno natančnostjo (f = 24-bitov)[4], razširjeno enojno natančnostjo (f = 32-bitov), dvojno natančnostjo (f = 53-bitov) in format z razširjeno dvojno natančnostjo (f = 64-bitov);
- pravila zaokroževanja števil s plavajočo vejico;
- računske operacije seštevanja, odštevanja, množenja, deljenja, kvadratnega korenjenja;
- obravnavo ostankov pri izvajanju računskih operacij;
- primerjavo števil;
- pretvorbo med različnimi formati zapisa števil s plavajočo vejico;
- pretvorbo med celoštevilčnim formatom in formatom s plavajočo vejico;
- zaokroževanje števil s plavajočo vejico na celoštevilčno vrednost;
- pretvorbo med osnovnimi formati zapisa števil s plavajočo vejico in decimalnim zapisom;
- izjeme in postopanje z njimi, vključno z obravnavo neštevil (NaNs).

Kljub navedenim, natančno določenim pravilom in formatom pa omenjena standarda ne določata [3], [5]:

- zgornje meje za število posebnih bitov[5], ki jih uporabljajo procesorji z razširjeno natančnostjo;
- natančnega zaokroževanja rezultata transcendentnih funkcij (eksponentnih, logaritemskih, trigonometričnih, inverznih trigonometričnih, hiperboličnih, inverznih hiperboličnih).

To je mogoče pričakovati različne rezultate enakih matematičnih operacij na različnih procesorjih. Oglejmo si podrobnosti izbranih testnih primerov in njihov namen:

1. testni primer

Algoritem: izračun rešitev logistične enačbe:

$$x_{n+1} = \lambda \cdot x_n \cdot (1 - x_n) \tag{5}$$

Za λ smo izbrali vrednost $\lambda = 3.8$, za začetno vrednost $x_0 = 0.2$ in število iteracij $n = 10^9$. Pri tako izbranih parametrih so rešitve logistične enačbe kaotične. Po 10^9 iteracijah se vrednost x zapiše v izhodno datoteko.

Namen: preveriti, ali bodo različni mikroprocesorji dajali enake rezultate pri množenju in odštevanju, ki ju standarda IEEE natančno določata.

[2] Obravnava eksponentni zapis števil z osnovo 2, npr. 2^x.
[3] Obravnava eksponentni zapis števil z osnovo 2 ali 10, npr. 2^x ali 10^x.
[4] f pomeni število bitov mantise
[5] Angl. Extra bits.

Ker je rešitev logistične enačbe kaotična, lahko zaradi znanega metuljevega efekta [7] pričakujemo, da se bo morebitna izredno majhna različnost procesorjev pri velikem številu iteracij enačbe 5 odrazila z zelo različnim končnim rezultatom enakega algoritma.

2. testni primer

Algoritem: izračun vsote kvadratnih korenov po enačbi:

$$y = \sum_{i=0}^{n} \sqrt{i} \; . \tag{6}$$

Pri tem smo za vrednost n izbrali veliko število $n = 10^9$. V izhodno datoteko se zapiše vrednost y po vsaki stoti operaciji seštevanja.
Namen: preveriti, ali bodo različni procesorji dajali enake rezultate pri operaciji korenjenja in seštevanja, ki ju standarda IEEE natančno obravnavata.

Iterativno korenjenje in seštevanje velikih števil smo izbrali zato, ker smo predvidevali, da bi lahko morebitno različnost procesorjev zaznali šele po zadostni akumulaciji odstopanj posameznega sumanda.

3. testni primer

Algoritem: izračun rešitev enačbe, ki vsebuje trigonometrični funkciji:

$$y_{n+1} = y_n + \sin(x) + \cos(x) \tag{7}$$

Pri tem je $y_0 = 0$, vrednosti x pa vzame iz intervala $134217728 \leq x \leq 134217728+2\cdot\pi$, po koraku 0.0000001. V izhodno datoteko se zapiše končna vrednost y.
Namen: preveriti, ali prihaja na različnih procesorjih do odstopanj pri računanju s trigonometričnimi funkcijami in velikimi argumenti, ki jih standarda IEEE 754 in IEEE 854 natančno ne določata.

Vsi trije algoritmi pomenijo rekurzivno izvajanje računskih operacij z namenom, akumulirati morebitna izredno majhna odstopanja, ki lahko tako postanejo dober indikator različnosti procesorjev.

3.2 Prevajanje izvorne kode testnega programa

Standarda IEEE 754 in IEEE 854 ne obravnavata, kako naj bi se ukazi višjeprogramskih jezikov prevedli v strojne oziroma procesorske ukaze [3]. Zato smo s testnim programom želeli preveriti tudi, ali obstaja možnost različne interpretacije istih ukazov višjeprogramskih jezikov na različnih procesorjih.

Zato smo izvorno kodo testnega programa prevedli[6] z dvema prevajalnikoma: Visual C++ 7.0 in GNUCC 3.2. S prvim smo kodo prevajali v načinu *Release* (končni) in *Debug*, z drugim pa v načinu *O3* in *Debug*. Dobili smo štiri različice istega testnega programa, ki smo jih nato inštalirali na računalnikih z različnimi procesorji.

S primerjavo rezultatov, ki smo jih dobili pri testiranju s posamezno različico izvršne kode, smo lahko sklepali o vplivu prevajalnika in načina prevajanja programa na njegovo obnašanje na računalnikih z različnimi procesorji.

3.3 Rezultati testiranja

Testiranje z opisanim testnim programom smo izvedli na računalnikih s procesorji: *AMD Duron*, *Intel Pentium IV*, *Intel Pentium MMX* in *Cyrix (IBM) 6x86MX*. Zanimalo nas je, ali je na teh računalnikih zagotovljena ponovljivost obnašanja testnega programa in ali lahko vrsta prevajalnika in načina prevajanja vplivata na ponovljivost rezultatov njegove izvršne kode.

Rezultate testiranja smo zbrali v tabeli 1. Oznaka NE pomeni, da rezultati v oklepaju navedenih testnih primerov (TP) niso ponovljivi, oznaka DA pa, da je bil kriterij ponovljivosti izpolnjen pri vseh treh testnih primerih.

Iz rezultatov razberemo, da so rezultati testnega programa:

1. ponovljivi, če njegovo izvorno kodo prevedemo s prevajalnikom GNUCC v načinu *O3* ali *Debug*;
2. ponovljivi samo na računalnikih s procesorji izdelovalcev *Intel* in *AMD*, če izvorno kodo testnega programa prevedemo s prevajalnikom Visual C++ v načinu *Debug*;
3. neponovljivi, če je izvorna koda prevedena s prevajalnikom Visual C++ v načinu *Release*;
4. neponovljivi, če je izvorna koda prevedena s prevajalnikom Visual C++ v načinu *Debug* in, če jih primerjamo med računalnikom s procesorjem *Cyrix* ter *Intel* ali *AMD*.

Ker so rezultati, ki smo jih dobili s testnim programom, prevedenim s pomočjo prevajalnika GNUCC, na vseh platformah popolnoma enaki, s pomočjo prevajalnika Visual C++ pa ne, sklepamo, da lahko na neponovljivo obnašanje programa vpliva tudi vrsta prevajalnika in način prevajanja izvorne kode. Ker je izvršna koda programa vedno popolnoma enaka, se lahko morebitna neponovljivost rezultatov pojavi le zaradi neenakosti v izvajanju ali interpretaciji iste strojne kode na primerjanih procesorjih.

[6] Izvedeno na računalniku s procesorjem *AMD Duron.*

Prevajalnik/ način prevajanja	Primerjani procesorji					
	Intel Pentium IV in *AMD Duron*	*Intel Pentium IV* in *Cyrix (IBM) 6x86MX*	*Intel Pentium IV in Intel Pentium MMX*	*Cyrix (IBM) 6x86MX* in *AMD Duron*	*Intel Pentium MMX* in *AMD Duron*	*Intel Pentium MMX* in *Cyrix (IBM) 6x86MX*
Visual C++ /Release	**NE** (TP 3)	**NE** (TP 3)	**NE** (TP 3)	**NE** (TP 3)	**NE** (TP 3)	**NE** (TP 3)
Visual C++ /Debug	**DA**	**NE** (TP1, TP2, TP3)	**DA**	**NE** (TP1, TP2, TP3)	**DA**	**NE** (TP1, TP2, TP3)
GNUCC/ O3	**DA**	**DA**	**DA**	**DA**	**DA**	**DA**
GNUCC /Debug	**DA**	**DA**	**DA**	**DA**	**DA**	**DA**

Tabela 1: Rezultati testiranja neponovljivosti s testnim programom, inštaliranim na računalnikih z navedenimi procesorji

Table 1: Results of non-repeatability testing using the test program installed on computers with the stated processors

3.4 Vzroki za neponovljivost

Rezultati testiranja kažejo, da je treba vzroke za neponovljivo obnašanje testnega programa iskati v procesorjih in/ali prevajalniku.

Razlogi za različne rezultate enakih matematičnih operacij v procesorjih so lahko naslednji:

- Različni procesorji ne podpirajo vseh formatov zapisa števil, ki jih obravnavata standarda IEEE 754 in IEEE 854. Primer: procesorji, kot so Intel x86/x87, Pentium, P6 in njegovi kloni proizvajalcev AMD in Cyrix, podpirajo primarno format z razširjeno dvojno natančnostjo in izvajajo vse aritmetične operacije z razširjeno dvojno natančnostjo, ne glede na velikost operandov, shranjenih v pomnilniku. Morebitno zahtevo po rezultatu z dvojno ali enojno natančnostjo izvedejo z ustrezno zaokrožitvijo rezultata z razširjeno dvojno natančnostjo. Zato lahko slednji odstopa od rezultata, dobljenega s procesorjem, ki podpira le računanje s števili z enojno ali/in dvojno natančnostjo.
- Lastnosti aritmetike s plavajočo vejico, ki jih standarda IEEE 754 in IEEE 854 natančno ne določata, so lahko v različnih procesorjih različno implementirane.
- Nekateri procesorji uporabljajo lastne posebne bite, ki v standardu IEEE 754 in IEEE 854 niso določeni [9]. Primer: DAZ (*Denormals Are Zeros*) bit in FZ (*Flush-to-zero*) bit pri procesorjih izdelovalca AMD [10]. Izdelovalci procesorjev poskušajo namreč z lastnimi dodatki doseči večjo atraktivnost na trgu.

Rezultati testiranja so pokazali, da lahko tudi vrsta prevajalnika in načina prevajanja vplivata na različno obnašanje programske opreme na različnih procesorjih. Razlog za to je lahko ena ali več napak v prevajalniku, zaradi katerih je dopuščena različna interpretacija istih procesorskih ukazov na računalnikih z različnimi procesorji.

4. Sklep

Neskladnost različnih procesorjev z določili standardov IEEE 754 in IEEE 854 in s standardi nedoločenih lastnosti procesorjev so lahko vzroki za neponovljivo obnašanje programske opreme. Izdelovalci procesorjev v izdelke vgrajujejo lastne dodatke, s katerimi želijo doseči večjo atraktivnost na trgu, čeprav njihov vpliv ni v skladu z obstoječimi standardi.

Neponovljivo obnašanje programske opreme pa je lahko tudi posledica napak v sami programski opremi ali v njenem prevajalniku. Glede na rezultate testiranja sklepamo, da prevajalnik Visual C++ v nasprotju s prevajalnikom GNUCC dopušča različno interpretacijo istih procesorskih ukazov na različnih procesorjih.

5. Literatura

[1] W. Kahan: Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic, Univerity of California, Berkeley, 1995, http://www.cs.berkeley.edu/~wkahan/ieee754status/.

[2] J. D. Darcy: Borneo 1.0.2 Adding IEEE 754 floating point support to Java, Berkeley, University of California, 1998.

[3] D. Goldberg: What Every Computer Scientist Should Know About Floating-Point Arithmetic, Computing Surveys, marec 1991.

[4] B. Verdonk, A. Cuyt, D. Verschaeren: A precision and range independent tool for testing floating-point arithmetic I: basic operations, square root and remainder, *ACM Transactions on Mathematical Software*, letnik 27, št. 1, 2001, str. 92−118.

[5] Sun Microsystems: Numerical Computation Guide, št. 806-3568-10, maj 2000.
[6] M. P. Kennedy: Three steps to chaos. II. A Chua's circuit primer, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, letnik 40, št. 10, oktober 1993, str. 657−674.
[7] M. Šalamon: *Problem neponovljivosti simulacij električnih vezij*, doktorska disertacija, 2003, UM FERI.
[8] M. Šalamon, T. Dogša: Problem neponovljivosti simulacij električnih vezij, *Inormacije. MIDEM*, 2004, letnik. 34, št. 1, str. 11−17.
[9] J. Demmel: Basic Issues in Floating Point Arithmetic and Error Analysis, http://www.cs.berkeley.edu/~demmel/cs267/lecture21/lecture21.html.
[10] Advanced Micro Devices: AMD64 Technology - AMD64 Architecture Programmer's Manual Volume 1: *Application Programming*, št. 24592, rev. 3.08, april 2003, str. 0−398

Matej Šalamon je docent na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru. Na raziskovalnem področju se ukvarja predvsem s kaotičnimi in kriptografskimi sistemi, simulatorji električnih vezij in njihovim testiranjem.

Tomaž Dogša je izredni profesor na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru, kjer predava na dodpilomski in podiplomski stopnji ter vodi Center za verifikacijo in validacijo sistemov. Na raziskovalnem področju se ukvarja predvsem s preverjanjem programske opreme in s simulatorji električnih vezij.