ELEKTROTEHNIŠKI VESTNIK 78(4): 165–170, 2011
EXISTING SEPARATE ENGLISH EDITION

Orodje za ovrednotenje vgrajenih sistemov na podlagi abstraktnega modeliranja sistema

Klemen Perko[1], **Tomaž Nahtigal**[2], **Andrej Trost**[2]

[1]*Sipronika d.o.o., Tržaška cesta 2, 1000 Ljubljana, Slovenija*
[2]*Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana, Slovenija*

E-pošta: andrej.trost@fe.uni-lj.si

Povzetek. Analiza in ovrednotenje sistema sta pomemben korak pri načrtovanju sodobnih vgrajenih sistemov. Od natančnosti pridobljenih rezultatov sta odvisna obseg in teža nadaljnjih odločitev v načrtovalskem postopku. V članku predstavljamo orodje ASyMod, s katerim ovrednotimo sistem že v prvih korakih načrtovanja. Orodje omogoča preprosto grafično modeliranje vgrajenih sistemov. Model vsebuje strojne in programske gradnike vgrajenega sistema na zelo abstraktni sistemski ravni. Abstraktne modele lahko hitro pripravimo in ocenimo izkoriščenost strojne opreme in časovni potek opravil. V članku bo prikazan primer izdelave in ovrednotenja abstraktnega modela vgrajenega sistema z orodjem ASyMod. Pokazali bomo, da se rezultati analize abstraktnega modela ujemajo z rezultati meritev na realnem vgrajenem sistemu.

Ključne besede: vgrajeni sistemi, sočasno načrtovanje strojne in programske opreme, načrtovanje na sistemski ravni

A design analysis tool for embedded systems based on abstract system modelling

Design analysis and evaluation are an important step in the embedded system design process. The accuracy of the evaluation results greatly affects decisions in the design process. The paper presents ASyMod, a tool to be used in evaluation of embedded systems in the early steps of the design process. The tool enables graphical composition of the system models. The embedded system models are composed of hardware and software components described on an abstract system level. The abstract models can be quickly assembled and evaluated according to the hardware usage and task execution timing. We will present an example of embedded system modelling and evaluation in the ASyMod tool. We will show that the abstract-model analysis results match with measurements on a real embedded system.

1 UVOD

Vgrajeni sistemi so elektronski sistemi, ki so vgrajeni v neko napravo in izvajajo funkcije nadzora, krmiljenja in obdelave podatkov. Sodobni vgrajeni sistemi so heterogene naprave, sestavljene iz več procesnih enot povezanih s komunikacijskimi vodili. Tipičen vgrajeni sistem opravlja večje število nalog, ki so razdeljene na opravila. Določena opravila se izvajajo na namenski strojni opremi (npr. vmesnik za komunikacijo po nekem protokolu), večina pa se izvaja v obliki programa na mikroprocesorjih [1]. Operacijski sistem omogoča, da se na mikroprocesorju v nekem času razvrsti večje število opravil, tako da je s stališča sistema videti, kot da se vsa opravila izvajajo hkrati.

Načrtovanje sodobnih vgrajenih sistemov je zaradi naraščanja kompleksnosti vse večji izziv. Zahteve po čim višji stopnji kakovosti, nizki ceni in čim hitrejši dostopnosti na trgu narekujejo uporabo sodobnih pristopov pri načrtovanju vgrajenih sistemov. Potrebujemo orodja, ki omogočajo zgodnje ovrednotenje sistema in posledično učinkovitejše vodenje pri načrtovalskih odločitvah. Pri načrtovanju vgrajenih sistemov je pomembno, da vnaprej ocenimo, kako se bodo opravila izvajala in kakšna bo izkoriščenost komponent strojne opreme [2]. Poleg tega pa imajo vgrajeni sistemi pogosto zelo natančno opredeljene časovne odzive, ki igrajo ključno vlogo pri načrtovalskih odločitvah. Na primer regulacijski sistem ima že v postopku načrtovanja določene periode vzorčenja in krmiljenja, pri katerih vsako odstopanje pomeni slabše regulacijske zmogljivosti.

Sodobni postopki načrtovanja na ravni sistemov (angl. System Level Design - SLD) [3], [4] uporabljajo modeliranje kot enega osnovnih konceptov. Postopek načrtovanja se začne z izdelavo abstraktnih modelov [5] brez podrobnosti, vezanih na implementacijo. Abstrakten model ovrednotimo glede na zahteve iz specifikacije. Če zahtevam iz specifikacije ne ustreza, model spreminjamo v postopku raziskovanja načrtovalskega prostora [6]. Ko so zahteve izpolnjene, model dopolnimo z novimi podrobnostmi in ga ponovno vrednotimo. To dopolnjevanje izvajamo, dokler v modelu ni dovolj informacij za implementacijo celotnega sistema.

V prispevku bomo predstavili orodje za modeliranje in ovrednotenje vgrajenih sistemov na zelo visoki ravni

Prejet 19. oktober, 2011
Odobren 7. november, 2011

abstrakcije. Definirali bomo osnovne elemente modela vgrajenega sistema, predstavili potek načrtovanja in rezultate ovrednotenja. Na primeru bomo pokazali, da se rezultati kljub zelo abstraktnemu modelu dobro ujemajo z meritvami na realnem sistemu.

2 ABSTRAKTEN MODEL VGRAJENEGA SISTEMA

Metodologija modeliranja na abstraktni ravni omogoča sočasno načrtovanje in ovrednotenje strojne in programske opreme [7]. Metodologija temelji na razširjeni uporabi abstrakcije, ki določa minimalen nabor podatkov, s katerimi je mogoče modelirati in ovrednotiti vgrajeni reaktivni sistem s porazdeljeno obdelavo podatkov. Pri reaktivnih sistemih je eden glavnih kriterijev čas izvajanja opravil, ki je odvisen od hitrosti in zasedenosti izvršilnih enot (npr. procesorjev) in komunikacijskih zakasnitev.

Načrtovanje reaktivnih sistemov začnemo na podlagi analize specifikacij z nekaj znanimi algoritmi, ki jih bo treba implementirati v strojni in programski opremi vgrajenega sistema. Algoritmi so sicer lahko že implementirani v nekem programskem jeziku, vendar je takšna implementacija samo ena od mogočih in zato ne nujno najboljša za začetne načrtovalske odločitve.

Model sistema razdelimo po znanem konceptu grafa Y [8] na opis *funkcionalnosti* (algoritmov)in *arhitekture* (strojne opreme) ter določimo medsebojno dodelitev.

Funkcionalnost določa naloge sistema in je opisana s pomočjo mreže abstraktnih opravil, povezane na podlagi medsebojnih podatkovnih odvisnosti. Abstraktni model opravila je opisan z zaporedjem visokonivojskih operacij za izvedbo izbranega algoritma in zahtev za prenos določene količine podatkov. Abstraktni model arhitekture je sestavljen iz množice izvršilnih in komunikacijskih enot. Izvršilne enote ponujajo storitve (izvedbo operacij, prenos podatkov), za katere je znana ocena časa, ki ga potrebujejo, da se izvedejo. Komunikacijske enote so namenjene modeliranju dostopnosti komunikacijskih kanalov (vodil). Več izvršilnih enot si lahko deli posamezno komunikacijsko enoto, vendar lahko istočasno do nje dostopa samo ena, preostale pa medtem čakajo.

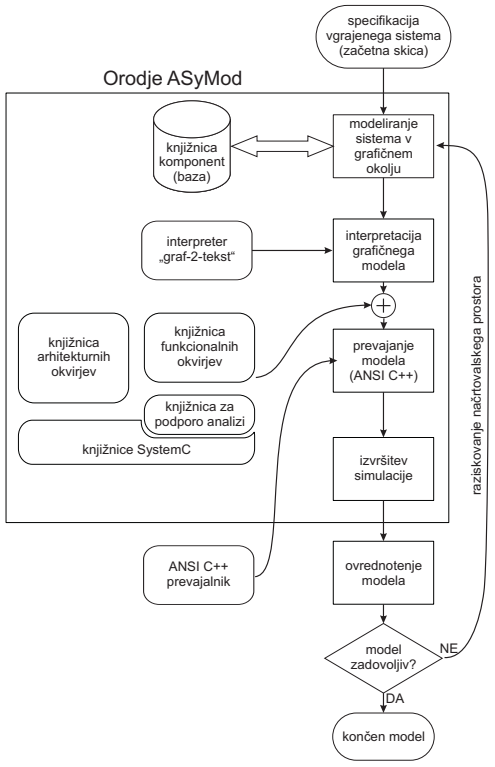
V abstraktnem opisu funkcionalnosti se nahajajo zahtevki za posamezne storitve, ki bodo omogočile izvedbo zahtevane funkcionalnosti. Za izvajanje teh zahtevkov je odgovorna tista izvršilna enota, kateri je v postopku *dodelitve* dodeljeno posamezno opravilo. Dodeljena izvršilna enota med simulacijo izvaja zahtevke skladno s svojo razpoložljivostjo in dostopnostjo z njo povezanih komunikacijskih enot.

Model izvršilnih enot omogoča posameznim opravilom ekskluzivni dostop, ki je model namenskih strojnih enot, ali pa prek razvrščevalnika, ki modelira delovanje operacijskega sistema na procesorju. Opravila so lahko v različnih stanjih:

- Wait - čaka na prožilni dogodek ali na komunikacijsko enoto,
- Ready - opravilo čaka na seznamu za izvajanje,
- Run - opravilo se trenutno izvaja.

Razvrščevalnik s seznama čakajočih opravil izbere tisto, ki naj se glede na svoje atribute izvaja, in se ga v seznamu označi kot trenutno izvajajoče se opravilo. Takrat začne opravilo posredovati svoje zahtevke za storitve. Ko se njegovo izvajanje konča, se s seznama briše. Model razvrščevalnika pozna tipične razvrščevalne algoritme, ki jih zasledimo v vgrajenih sistemih [9]: FCFS (angl. First-come first-served), RM (angl. Rate Monotonic), DM (angl. Deadline Monotonic) in EDF (angl. Earliest Deadline First). Poleg tega je mogoče izbrati prekinjanje izvajanja opravil, ki imajo nižjo prioriteto kot tista ki so na čakalnem seznamu.

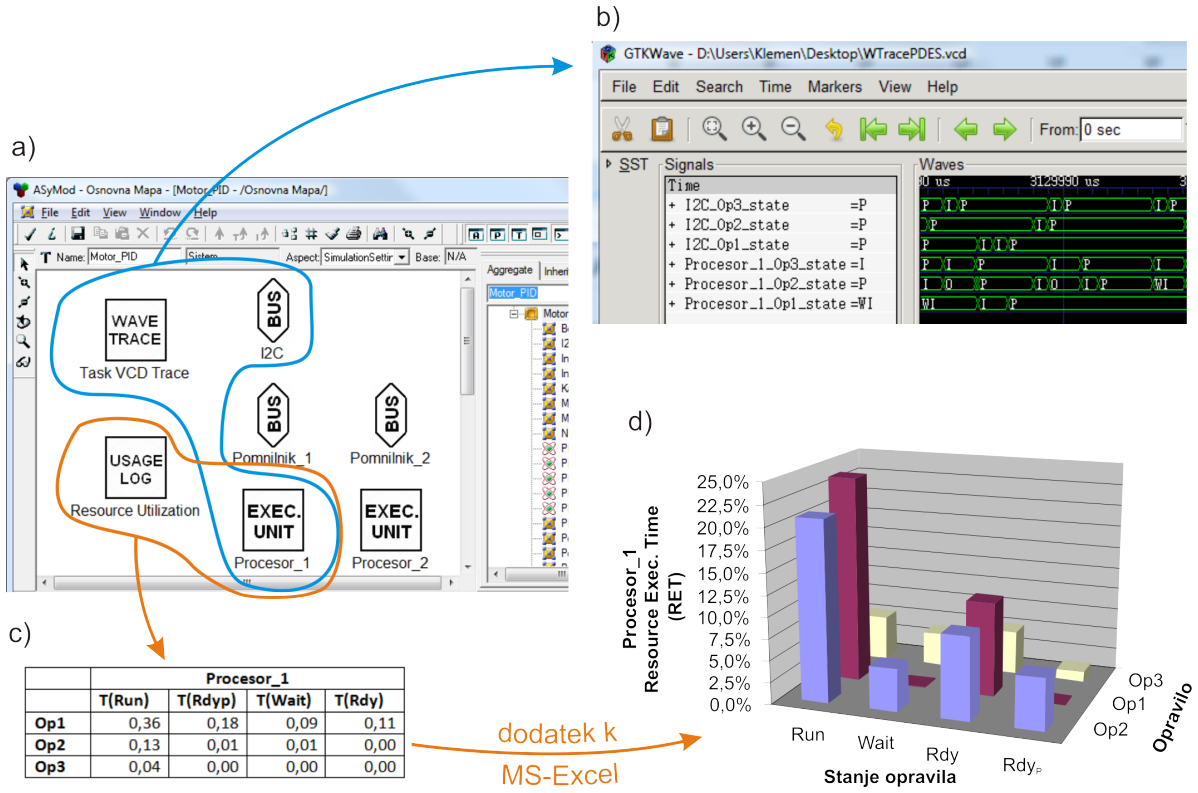
3 ORODJE ZA MODELIRANJE SISTEMOV - ASYMOD



Slika 1: Potek modeliranja vgrajenih sistemov v orodju ASyMod

Razvili smo orodje za izdelavo in ovrednotenje abstraktnih modelov vgrajenih sistemov, ki smo ga poimenovali ASyMod (Abstract System-level Modelling). Orodje je sestavljeno iz grafičnega modelirnega okolja s podpornimi knjižnicami in prevajalnikom ter skriptami za ovrednotenje modelov.

Slika 1 prikazuje gradnike orodja ASyMod in potek načrtovanja vgrajenih sistemov. Načrtovalec ima na voljo bazo vnaprej pripravljenih komponent v grafičnem



	Procesor_1			
	T(Run)	T(Rdyp)	T(Wait)	T(Rdy)
Op1	0,36	0,18	0,09	0,11
Op2	0,13	0,01	0,01	0,00
Op3	0,04	0,00	0,00	0,00

Slika 2: Ovrednotenje modela vgrajenega sistema v orodju ASyMod: a) grafično modelirno okolje, b) simulacija časovnega poteka opravil, c) tabelarični izpis izkoriščenosti enot in d) grafikoni stanj opravil

vmesniku. Grafični modelirni vmesnik smo naredili v prosto dostopnem modelirnem okolju GME (Generic Modeling Environment) [10]. Okolje GME smo za naš namen prilagodili z določitvijo metamodela, ki vsebuje vse sintaktične, semantične in predstavitvene informacije.

Grafično modeliranje poteka v treh pogledih (angl. aspect): *arhitekturnem*, *funkcionalnem* in *dodelitvenem*. V *arhitekturnem* in *funkcionalnem* se definirajo arhitekturni in funkcionalni elementi modela. Posamezni elementi se lahko pridobijo iz obstoječe *knjižnice komponent*, oz. se izdelajo na novo in nato shranijo v knjižnico za poznejšo ponovno uporabo. Elementi morajo biti zgrajeni tako, da ustrezajo podpornim knjižnicam. Tu uporaba grafičnega okolja razbremeni uporabnika, saj ga usmerja pri izdelavi posameznega elementa, s sprotnim preverjanjem pravil pa onemogoča izdelavo neustreznih elementov. V *dodelitvenem* pogledu se nato izvede dodelitev arhitekturnih k funkcionalnim elementom.

Po končanem koraku grafičnega modeliranja sistema sledi interpretacija grafičnega modela. Razvili smo prevajalnik, ki iz grafičnega modela zgradi opis v jeziku za načrtovanje na ravni sistemov SystemC. Rezultat interpretacije je datoteka z abstraktnim opisom vgrajenega sistema in simulacijskimi nastavitvami (npr: časovni korak, trajanje simulacije itd.)

V naslednjem koraku sledi prevajanje jezikovnega opisa s prevajalnikom ANSI C++ v izvršljivo datoteko. Ko izvršimo datoteko, dobimo rezultate simulacije abstraktnega sistema v obliki časovnih potekov opravil in poročila o izkoriščenosti arhitekturnih sredstev in grafikona, kot prikazuje slika 2.

Ovrednotenje modela vgrajenega sistema naredimo s statistično obdelavo simulacijskih rezultatov. V tabelaričnem izpisu so za vsako opravilo podatki o deležu časa izvajanja, čakanja na izvajanje, čakanja na podatke in časovnem deležu, ko je bilo prekinjeno. Za analizo reaktivnih sistemov so pomembne tudi časovne variacije periodičnih opravil, ki jih razberemo iz histogramov, kot bo prikazano na primeru.

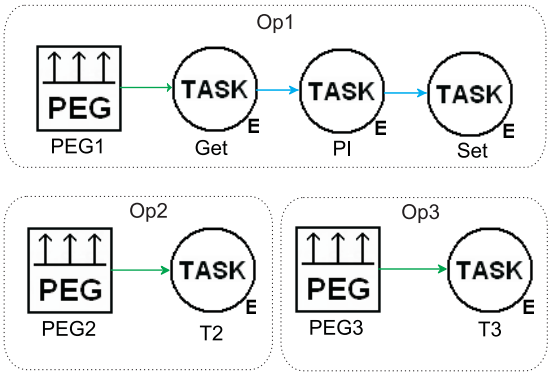
Načrtovanje vgrajenih sistemov je iterativni postopek. Če prvotni model ni zadovoljiv, ga popravimo in ponovno ovrednotimo. Ta postopek raziskovanja načrtovalskega prostora ponavljamo, dokler ne pridemo do zadovoljivega modela.

4 PRIMER MODELIRANJA VGRAJENEGA SISTEMA

Uporabo orodja ASyMod bomo predstavili na abstraktnem modelu sistema za vodenje hitrosti vrtenja enosmernega motorja. Osnovna naloga sistema je periodično izvajanje diskretnega regulacijskega algoritma, ki ga lahko razdelimo na tri podatkovno odvisna opravila: branje senzorja (`Get`), izračun odziva (`PI`) in nastavljanje aktuatorja (`Set`), kot prikazuje slika 3. Na splošno

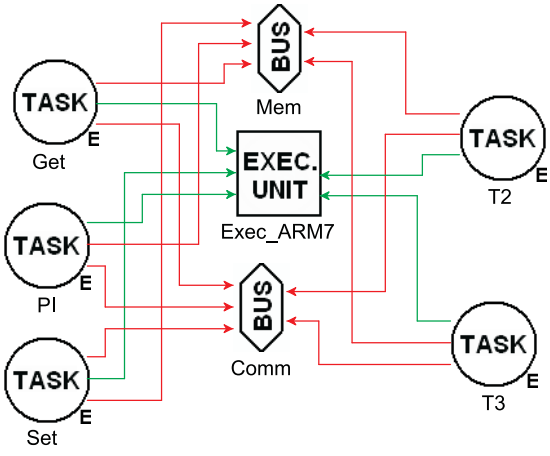
izvaja vgrajeni sistem še druge funkcionalnosti, ki smo jih modelirali kot dve novi periodični opravili: `T2` in `T3`. Dodatni opravili sta podatkovno neodvisni od regulacijskega, vendar delita z njim skupno izvršilno in komunikacijsko enoto, zato pričakujemo da bosta vplivali na delovanje sistema.

Ker imamo poleg zaporednih podatkovno odvisnih opravil še dve neodvisni, ki delita skupno izvršilno enoto, uporabimo operacijski sistem z razvrščevalnikom. Razvrščevalniku smo dodelili tri opravila: `Op1`, `Op2` in `Op3`, ki so obkrožena na sliki 3. Izbrali smo razvrščevalni algoritem RM v prekinjevalnem načinu.



Slika 3: Funkcionalni pogled na model v orodju ASyMod

Arhitekturni del vsebuje izvršilno enoto `Exec_ARM7` in komunikacijski enoti `Mem` in `Comm`. Prvi dve enoti sta procesorska enota in pripadajoči pomnilnik. Enota `Comm` pa pomeni zaporedno komunikacijsko vodilo med vgrajenim sistemom in motorjem s senzorjem vtljajev. Na sliki 4 je prikazan dodelitveni pogled modela v orodju ASyMod.



Slika 4: Dodelitveni pogled na model v orodju ASyMod

Slika 5 prikazuje visokonivojsko kodo, ki opisuje opravila `Get` in `PI` in izsek kode iz knjižnice izvršilne enote ARM7. Opis opravila `Get` vsebuje zahtevek za prenos dveh bajtov iz komunikacijske enote `Comm`.

Za izdelavo modela potrebujemo oceno časa za prenos določene količine podatkov po vodilu in časa za izračun

```
// Abstrakten opis opravil v ASyModu
void Get::MainThread()
{
 m_pExecUnit->GetData(this, 2, &Comm, -1);
}

void PI::MainThread()
{
 m_pExecUnit->Calc(this, 563);
}

// Storitve iz knjižnice ARM7
void Exec_ARM7::Calc(int n)
{
  for (int i=0; i<563; i++)
  {
    Wait(this, sc_time(64, SC_NS));
  }
}
```

Slika 5: Visokonivojski model v jeziku SystemC

določenega algoritma. Čase določimo z nizkonivojsko simulacijo ali eksperimentalno izmerimo in jih modeliramo v obliki storitev v knjižnici določene izvršilne enote. Za procesor ARM7 smo ocenili čas izvajanja regulacijskega algoritma na 36 µs, kar pomeni 563 strojnih inštrukcij. Ker se lahko opravilo `PI` kadarkoli prekine, je njegov model opisan z zanko, ki se ponovi 563-krat in vsakokrat čaka 64 ns, kolikor je povprečen čas izvajanja inštrukcij.

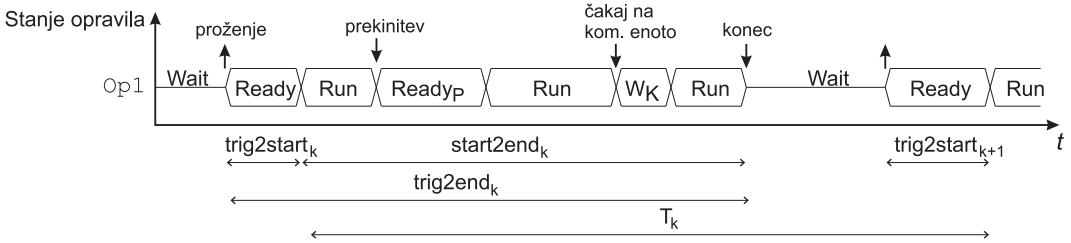
5 Rezultati

Iz časovnih potekov izvajanja opravil lahko razberemo nekaj karakterističnih časov pri vsakokratnem izvajanju opravila. Na sliki 6 so za k-to zaporedno izvajanje opravila označeni naslednji štirje časi:

1) $trig2start_k$: je čas, ki je pretekel od takrat, ko je prišel zahtevek za izvedbo opravila, pa do takrat, ko se je opravilo začelo izvajati,
2) $start2end_k$: je čas, ki je pretekel od začetka pa do konca izvajanja; v ta čas se štejejo tudi vse prekinitve izvajanja,
3) $trig2end_k$: je pretečeni čas, od zahtevka za izvedbo pa do konca izvajanja opravila,
4) T_k: je čas med posameznima začetkoma izvajanja opravila.

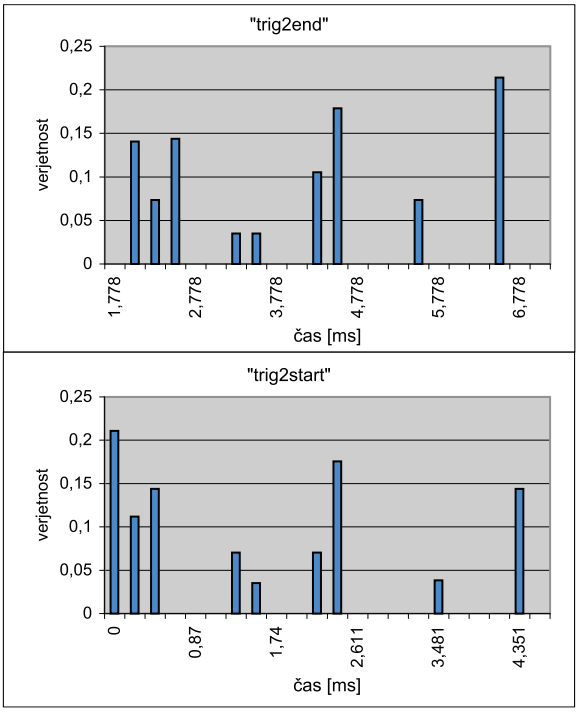
Skozi celoten časovni potek za posamezno opravilo lahko te čase zberemo skupaj, jih statistično obdelamo in prikažemo na histogramu.

Na sliki 7 so normalizirani histogrami, ki prikazujejo porazdelitev dveh karakterističnih časov opravila `Op1`. Populacija vzorcev v histogramih znaša 313 meritev, razdeljeni pa so v 22 stolpcev. Porazdelitve v vseh histogramih so diskretne. Opravilo `Op1` ima med vsemi opravili določeno najnižjo prioriteto. Zakasnitev izvajanja opravila $trig2start$ je najbolj porazdeljena. Verjetnost, da se bodo opravila lahko takoj začela izvajati



Slika 6: Karakteristični časi pri posameznem izvrševanju opravila.

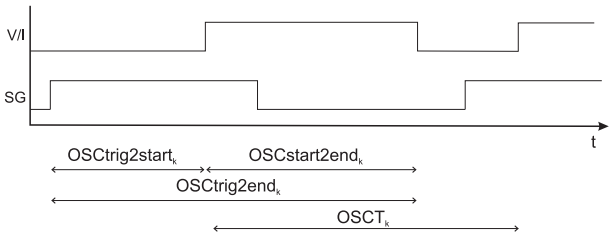
komaj presega vrednost $p_1 = 0,2$, zato je precej velika verjetnost, da bo opravilo moralo počakati, da se konča eno ali celo obe opravili, ki imata višjo prioriteto.



Slika 7: Histogrami karakterističnih časov opravila `Op1`

Karakteristične čase izvajanja opravil smo tudi izmerili na implementiranem sistemu. V programsko opremo smo dodali testno kodo, ki ob začetku izvajanja opravila postavi izhodni priključek mikrokrmilnika na logično 1, ob koncu izvajanja pa na 0. Meritve smo opravili s pomočjo digitalnega osciloskopa Lecroy WaveRunner in referenčnega signalnega generatorja. Funkcije digitalnega osciloskopa omogočajo kontinuirano shranjevanje rezultatov posameznih meritev in prikaz vseh shranjenih rezultatov v obliki histograma, kot prikazuje slika 8.

Izmerjena širina impulza $OSCstart2end_k$ je enaka karakterističnemu času $start2end$. $OSCT_k$ je izmerjeni čas med dvema pozitivnima frontama signala na izhodnem prilkjučku in je enak periodi opravila T. Za meritve karakterističnih časov $trig2start$ in $trig2end$ smo potrebovali še referenčni signalni generator. Temu smo nastavili, da je generiral signal z enako periodo



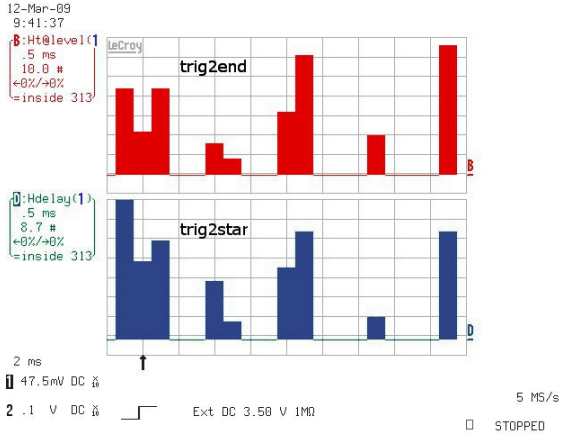
Slika 8: Meritve karakterističnih časov opravil z osciloskopom.

kot opravilo, ki smo ga opazovali. Prva fronta signala je osciloskopu prožila vzorčenje signala na priključku mikrokrmilnika. Meritve časa $OSCtrig2start_k$ med proženjem in prvo fronto impulza na izhodnem priključku so v korelaciji s časom $trig2start$. Enako so meritve časa $OSCtrig2end_k$ med proženjem in zadnjo fronto impulza pa v korelaciji s časom $trig2end$.

Pomanjkljivost meritev $OSCtrig2start$ in $OSCtrig2end$ je v tem, da je med signalom referenčnega generatorja in taktom notranjega časovnika, ki proži izvajanje opravila neki fazni zamik. Izmerjena časa sta od dejanskih časov $trig2start$ in $trig2end$ povečana ravno za velikost tega zamika. Zamik lahko pred začetkom meritev ročno zmanjšamo tako, da za nekaj časa nekoliko spremenimo periodo referenčnega signalnega generatorja, vendar pa ga tako ne moremo sistematično odpraviti. Ker imajo vsi izmerjeni časi enako napako, velja, da nenatančnost zaradi zamika ne vpliva na obliko histograma (višino stlpcev in njihovo medsebojno razdaljo), ampak samo na horizontalni premik vseh stolpcev. Čas, ki ga pomeni vsak stolpec histograma, je podaljšan za čas zamika. Na sliki 9 sta prikazana histograma časov $OSCtrig2end$ in $OSCtrig2start$. Hitra vizualna primerjava oblike teh dveh histogramov nam pove, da sta zelo podobna histogramom simulacij iz ASyModa (slika 7).

Histograme smo kvantitativno primerjali z metodo preseka normiranih histogramov. Uporabljena mera za podobnost dveh normaliziranih histogramov, pri čemer imata oba B stolpcev, je definirana kot seštevek presekov istoležnih stolpcev. Mera SIM je podana z enačbo (1).

$$SIM = \sum_{i=1}^{B} min(p_i, p_{Ri}) \quad . \tag{1}$$



Slika 9: Histograma meritev $OSCtrig2start$ in $OSCtrig2end$ za opravilo `OP1`

Rezultat mere se giblje $0 \leq SIM \leq 1$, pri čemer velja, da bolj ko sta si histograma podobna, bliže je rezultat vrednosti 1. Primerjava histogramov $trig2end$ za opravilo `OP1` da rezultat $SIM = 0.97$, $trig2start$ pa rezultat $SIM = 0.99$.

6 Sklep

Z razvojem orodja ASyMod za modeliranje vgrajenih sistemov na sistemski ravni smo postavili dobra izhodišča za nadaljnje raziskovalno delo na tem področju. Orodje je osnova za modeliranje sistemov na zelo visokem ravni abstrakcije. Možnosti za nadaljnje delo vidimo v več smereh.

Trenutno je ASyMod primeren za modeliranje sistemov, pri katerih je čas izvajanja funkcionalnih opravil neodvisen od vhodnih podatkov. Smiselno bi bilo razširiti možnosti modeliranja tako, da bi se upoštevala tudi ta odvisnost. Obenem bi bilo smiselno vključiti še opazovanje časa, ki je posledica preklopa konteksta (angl. context switch) pri izvajanju opravil.

Naslednja možnost za razširitev se kaže pri konceptu komunikacijskih enot. Trenutno so te enote modelirane s konceptom mutexa – enote so ali pa niso zasedene, trajanje dostopa pa je modelirano znotraj opisa funkcionalnosti. Obstaja pa še drug pogled na komunikacijske kanale. Različni kanali imajo različno prepustnost. Če želi funkcionalnost samo prenesti neko sporočilo, naj od komunikacijske enote zahteva storitev za ta prenos. Samo od zmogljivosti komunikacijske enote naj bo odvisno, koliko časa bo potrebovala za prenos tega sporočila. Tako bi bolje razmejili funkcionalne zahteve sistema od zmožnosti, ki jih ponujajo arhitekturni elementi.

Literatura

[1] Frank Vahid and Tony Givargis, “Embedded System Design: A Unified Hardware/Software Introduction”, John Wiley and Sons, 2002.

[2] J. Kreku, M. Hoppari, T. Kestilä, Y. Qu, J.-P. Soininen, P. Andersson and K. Tiensyrjä, “Combining UML2 application and SystemC platform modelling for performance evaluation of real-time embedded systems”, EURASIP Journal on Embedded Systems, vol. 2008, no. 6, pp. 1 - 18, 2008.

[3] V. Zivkovic and P. Lieverse, “An Overview of Methodologies and Tools in the Field of System-Level Design”, Embedded Processor Design Challenges, Springer-Verlag New York, Inc., pp. 74 - 88, 2002.

[4] Alberto Sangiovanni-Vincentelli, “Quo Vadis SLD: Reasoning about Trends and Challenges of System-Level Design”, *Proceedings of the IEEE*, vol. 95, no. 3, pp. 467 - 506, March 2007.

[5] D.C. Schmidt, “Guest editor’s introduction: model-driven engineering”, IEEE Computer, vol. 39, no. 2, pp. 25 - 31, 2006.

[6] A. Pimentel, “The Artemis Workbench for System-level Performance Evaluation of Embedded Systems”, Journal of Embedded Systems, vol. 3, no. 3, pp. 181 - 196, 2008.

[7] J. Dedič, M. Finc, A. Trost, “A methodology for supporting system-level design space exploration at higher levels of abstraction”, J. Circ. Syst. Comput., vol. 17, no. 4, pp. 703 - 727, 2008.

[8] Daniel D. Gajski and Robert H. Kuhn, “New VLSI tools” *Computer*, vol. 16, no. 12, pp. 11 - 14, 1983.

[9] G. Buttazzo, “Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications”, Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[10] A. Ledeczi et all, “The Generic Modeling Environment”, Proceedings of IEEE Int. Workshop on Intelligent Signal Processing (WISP ’01), IEEE, Budapest, Hungary, maj 2001.

Klemen Perko je diplomiral in doktrorial na Fakulteti za elektrotehniko Univerze v ljubljani v letih 2004 in 2011. Od leta 2005 je zaposlen v podjetju Sipronika, d.o.o., Ljubljana, kjer izvaja številne aplikativne projekte. Njegovo raziskovalno delo je usmerjeno na visokonivojsko načrtovanje in modeliranje vgrajenih sistemov.

Tomaž Nahtigal je doktorski študent na Fakulteti za elektrotehniko Univerze v Ljubljani. Diplomiral je v letu 2007. Ukvarja se s postopki načrtovanja in verifikacije digitalnih sistemov, aplikacij in sočasnim načrtovanjem strojne in programske opreme.

Andrej Trost je doktoriral leta 2000 na Fakulteti za elektrotehniko Univerze v Ljubljani in bil habilitiran v naziv docent. Raziskovalno se ukvarja z razvojem vezij v tehnologiji FPGA in načrtovanjem digitalnih sistemov, ki jih razvija tudi za industrijske aplikacije. Visokonivojsko načrtovanje vezij poučuje pri predmetih na dodiplomskem in podiplomskem študiju.