REALIZACIJA MODULA ZA NADZOR REAL-TIME PROCESA U VHDL-u

Sandra Brankov, Milun Jevtić, *Elektronski fakultet u Nišu*

Sadržaj – *U radu će biti predstavljen jedan modul za nadzor real-time procesa i njegova realizacija u VHDL-u. Ovaj modul omogućava nadzor vremenskih karakteristika u procesu testiranja sistema kao i kontinualno on-line proveravanje svih vremenskih karakteristika sistema u toku rada.*

1. UVOD

Osnovna karakteristika po kojoj se razlikuju real-time sistemi (sistemi koji rade u realnom vremenu) od ostalih sistema jeste ponašanje u vremenu. Da bi real-time sistem (RTS) ispravno funkcionisao potrebno je ne samo da daje korektan rezultat na izlazu već i da ga da u tačno definisanom vremenskom intervalu, [1]. Ovo je posebno kritično kod hard real-time sistema (HRTS) jer neblagovremeno izvršavanje procesa može dovesti do katastrofe.

U postupku projektovanja RTS-a često se prave pretpostavke o ponašanju sistema i njegove okoline, a posebno su važne one pretposatvke koje se odnose na vremenska ograničenja. Vremenska ograničenja se definišu u skladu sa datim karakteristikama procesa i sistema tj. vremenskim rokovima zadataka, prioritetima i dostupnim resursima, kao i u skladu sa nepredviđenim spoljašnjim događajima tj. uticajem okoline. Osnovni problem koji se javlja pri projektovanju sistema za rad u realnom vremenu jeste dokazivanje korektnog zadovoljavanja vremenskih ograničenja. Zbog toga se javlja potreba za nadzorom (monitoring) procesa i događaja unutar RTS-a u realnom vremenu, pri čemu aplikaciju za rad u realnom vremenu obično čini skup kooperativnih zadataka, odnosno procesa. Nadzor u realnm vremenu predstavlja jedan od alata za testiranje i debagovanje RTS-a.

U ovom radu će biti predstavljen jedan modul za nadzora real-time procesa i njegova realizacija u VHDL-u. Ovaj modul omogućava nadzor vremenskih karakteristika tokom testiranja sistema kao i kontinualno on-line proveravanje svih vremenskih karakteristika sistema u toku rada.

2. NADZOR U REALNOM VREMENU

Cilj nadzora u realnom vremenu je da sačuva performanse sistema u opsegu u kome se ne menja redosoled i tajming događaja. Nadzor je značajan za identifikaciju i rešavanje sledećih problema [2]:

1. Okolina izvršenja za mnoge sisteme je nesavršena i interakcija sa spoljašnjim svetom uvodi dodatnu nepredvidivost;
2. Pretpostavke pri projektovanju mogu biti narušene u toku rada sistema zbog neočekivanih uslova;
3. Primena formalnih tehnika i algoritama vremenskog planiranja zahteva pretpostavke o osnovnom sistemu;
4. Može biti neostvarivo verifikovati formalno neka svojstva RTS-a tokom logičkog projektovanja.

Dakle, neophodne su provere u toku rada sistema za dokazivanje korektnog funkcionisanja sistema u realnom vremenu. To važi kako kod procesa projektovanja nakon faze integracije, tako i u toku eksploatacije sistema radi povećane pouzdanosti i predvidivog ponašanja sistema prema okolini.

Nadzor izvršenja procesa u realnom vremenu može se razmatrati kao alat za testiranje i debagovanje sistema. U procesu testiranja funkcija nadzora služi da pravi arhivu događaja o vremenskim karakteristikama procesa u sistemu. Može se, na primer, pamtiti trenutak kada procesi počinju sa izvršenjem i trenutak kada završavaju. Zatim, ako se uzmu u obzir gornje i donje granice za izvršenje procesa kao i WCET (Worst Case Execution Times) [1] karakterističnim za svaki proces, vrše se provere da li su se procesi izvršili u skladu sa primenjenim algoritmom za izvršenje zadataka ili ne. Ako se jave neke nepravilnosti moguće ih je još u fazi testiranja otkloniti tj. korigovati sistem ili primenjeni algoritam.

Postoje tri široka pristupa realizacije nadzora: softverski, hardverski i hibridni pristup.

U okviru softverskog pristupa nadzoru postoje dve softverske tehnike nadzora:

(1) Jedna tehnika ugrađuje nadzorni kod u ciljni program i ona nije transparentna.

(2) Druga tehnika ugrađuje nadzorni kod unutar sistemskog kernela ili tretira monitorski kod kao proces odvojen od procesa ciljnog koda. Ovaj metod je transparentan ali manje fleksibilan.

Softverski nadzor, koji zahteva detekciju događaja i sakupljanje podataka o događajima, može se realizovati na sledeće načine:

- Ciljni program detektuje događaje i sakuplja podatke o događajima.
- Ciljni program detektuje događaje a nadzor sakuplja podatke o događajima.
- Kernel detektuje događaje a nadzor sakuplja podatke o događajima.

Sve tri implementacije imaju svoje prednosti i nedostatke. Implementacija oba, i detekcije događaja i sakupljanja događaja, unutar ciljnog programa je najlakši i najdirektniji način za nadzor sistema. Ostvarivanje detekcije događaja i prikupljanja događaja na nivou kernela ima dve prednosti u odnosu na ono na programskom nivou: jedno je transparentnost, a drugo redukovanje nadzornih perturbacija (smetnje, uznemirenje). Transparentnost se lako postiže jer korisnici ne moraju da modifikuju ciljne programe zbog detekcije događaja i prikupljanja događaja. Perturbacije su

[*] *Rad je (delimično) finansiran od MNTR u okviru projekta IT.1.05.0077.A - Upravljacko nadzorni sistemi za rad u realnom vremenu*

redukovane jer je promena konteksta između ciljnog programa i kernela izbegnuta. Nasuprot tome, izvršavanje detekcije događaja na programskom nivou zahteva sistemski poziv. Ovo rezultuje u ekstenzivnoj promeni konteksta jer se sistemski pozivi implementiraju sa interaptima kernelu. Međutim, mada pristup na nivou kernela redukuje perturbacije, dodatni stalni nadzor na kernelu ipak usporava vreme izvršenja kernela i povećava vreme izvršenja ciljnog programa.

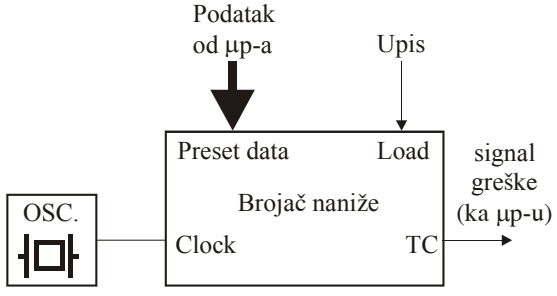
Hardverski pristup zahteva mernu hardversku platformu na kojoj se aplikacija odvija. Koristi se specijalizovani hardver za lečovanje podataka direktno iz unutrašnjih basova ciljnog sistema. Hardverski pristupi imaju prednost nenametljivosti, ali generalno obezbeđuju veoma nizak nivo podataka.

Generalno, hardverski nadzor se koristi da bi se monitorovalo ponašanje u toku izvršenja ili hardverskih elemenata ili softverskih modula. Prvo se koristi kod merenja performansi hardverskih elemenata, kao što je merenje pristupa kešu, vreme pristupa memoriji, totalno CPU vreme, totalno vreme izvršenja, I/O zahtevi, I/O dozvole i vreme zauzetosti I/O uređaja. Nadzor softverskih modula koristi se kod debagovanja i kod analize performansi takvih softverskih karakteristika kao što su: programska uska grla, mrtve petlje i stepen paralelizma. Izgleda da su ove dve upotrebe nadzora sasvim različite, ali rezultati merenja hardverskih performansi mogu takođe da pomognu kod analize softverskih performansi i debagovanja.

Hibridna realizacija sistema nadzora daje zajedno nenametljivu prirodu hardverskih pristupa i fleksibilnost softverskih pristupa realizacije nadzora.

Hibridni pristup sastoji se od softverske implementacije i hardverske detekcije, dakle, komponenta između hardverskog i softverskog nadzora. Ciljni program koji se nadzire prvo se instrumentalizuje ručno ili automatski da bi generisao događaje, a zatim se hardverski nadzor koristi za detekciju događaja i prikupljanje odgovarajućih podataka o događajima. Hardverski nadzor može se projektovati kao stalni deo ciljnog sistema u toku faze projektovanja i u ovom slučaju nadzor se zasniva na ispitivanju signala na basovima ciljnog sistema, ili on može biti individualni element, odnosno koprocesor integrisan u ciljni sistem u toku faze testiranja i debagovanja.

U procesu nadzora za detekciju toka kontrole koriste se nadzorni tajmeri (Watchdog) koji proveravaju vreme izvršenja programa. Nadzorni tajmeri restartuju sekvence koje su ugrađene u aplikacionom programu i inicijaliziraju periodično narzorni tajmer. Ako nisu inicijalizirani (zbog greške) unutar specificiranog vremena, nadzorni tajmeri generišu signale greške.



Sl. 1. *Nadzorni tajmer*

Pretpostavimo da nadzorni tajmer proverava donju ($T-T_w$) i gornju (T) granicu vremena izvršavanja (t_i) programskog segmenta (CS_i) a parametri T i Tw mogu biti fiksni za sve segmente. Može se proceniti verovatnoća p_{ij} (T_w) da će nadzorni tajmer otkriti pogrešnu granu segmenta CS_i u segmentu CS_j (sa vremenom izvršavanja t_j):

$$p_{ij}(T_w) = [(t_i + t_j - T)^2 + (T - T_w)^2] / (2\, t_i\, t_j) .$$

Optimalna situacija je kada je $t_i=t_j=T-T_w/2$. Smanjivanjem T_w, možemo postići sasvim dobro pokrivanje greške. Međutim ovo zahteva znanje o vremenu izvršavanja programa (procenjeno specijalnim programima). Najpopularniji nadzorni tajmeri proveravaju samo gornju granicu ($T_w=T$). U ovom slučaju, maksimalana vrednost od p_{ij} ($T_w=T$) iznosi 0,5. Jedan takav nadzorni tajmer prikazan je na slici 1.

U procesu nadzora često se koristi mehanizam prekida koji je standardna funkcija skoro svih savremenih mikroprocesora. Prekidi se koriste za dojavu asinhronih događaja koji se javljaju unutar mirkroprocesorskog sistema ili u njegovom okruženju, a koji zahtevaju odgovarajuću programsku obradu. Tipično, zahtev za prekid inicira periferni uređaj tražeći, na taj način, da bude "opslužen" od strane mikroprocesora. Prekidni potprogrami su obično kraće programske sekvence kojima se "opslužuje" uređaj ili kolo koje je iniciralo prekid (prenos primljenog podatak iz serijskog U/I interfejsa u memoriju, "skaniranje" tastature da bu utvrdio koja dirka je pritisnuta…). Nakon završetka prekidnog potprograma, mikroprocesor nastavlja izvršenje prekinutog programa. Tipično, mikroprocesori poseduju mali broj linija za prihvatanje zahteva za prekid tako da se problem javlja u situacijama kada u mikroprocesoru postoji više "izvora" zahteva za prekid od broja raspoloživih "prekidnih" linija mikroprocesora. Ovaj problem se rešava ugradnjom specijalizovanog kola, tzv. programabilnog kontrolera prekida.

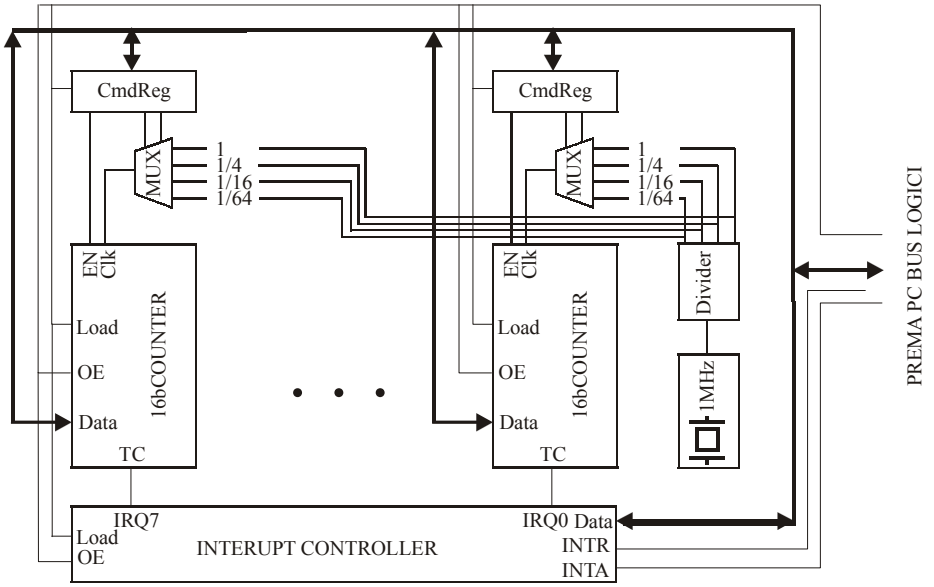
Zadatak kontrolera prekida je da zahtev za prekid upućen od strane nekog od perifernih uređaja (signali IRQ0-7) prosledi mikroprocesoru (signal INT) i da nakon prihvatanja zahteva za prekid od strane mikroprocesora (aktivan je signal INTA), preko sistemske magistrale preda mikroprocesoru informaciju o tome koji od uređaja je inicirao prekid. Pored toga, programabilni kontroler prekida omogućava selektivnu dozvolu/zabranu pojedinih zahteva za prekid (tj. maskiranje prekida), pridruživanje prioriteta svakom od zahteva za prekid (ako je više od jednog uređaja izdalo zahtev za prekid, prihvata se zahtev najvišeg prioriteta).

3. MODUL ZA NADZOR REAL-TIME PROCESA

Na slici 2. predstavljeno je jedno hardversko rešenje nadzora RTS-a koje omogućava nadzor vremenskih karakteristika za vreme debagovanja i testiranja sistema u samom procesu projektovanja kao i kontinualno on-line proveravanje svih vremenskih karakteristika u toku rada [3]. Ovaj modul za nadzor realizovan je korišćenjem hibridne tehnike za realizaciju nadzora.

Brojači-tajmeri koriste se kao uređaji za definisanje vremenskih trenutaka pojavljivanja događaja i kao watchdog tj. nadzorni tajmeri za proveru korektnog vremenskog izvršavanja zadataka. Komandni registar, CmdReg, vrši izbor kvanta od 1, 4, 16 ili 64 μs odnosno maksimalno vreme za

izvršenje zadataka 65.5, 262, 1048 i 4194 ms. Komandni registar, preko EN-enable ulaza dozvoljava tj. zabranjuje rad brojača. U okviru modula za nadzor nalazi se i jedan dekoder koji na osnovu ulaznih podataka propušta signal za dozvolu upisa odnosno signal za dozvolu čitanja, do određenog komandnog registara odnosno brojača. Preko linije za prenos podataka Data vrši se upis u komandni registar i u brojač.



Sl. 2. *Hardversko rešenje nadzora RTS-a*

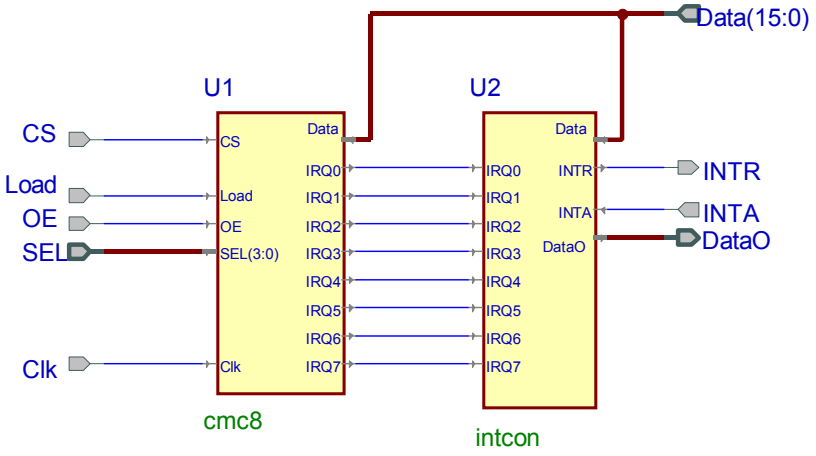
Prvo se upisuje-postavlja brojač na željenu vrednost a zatim se izvrši upis u komandni registar ako ima promena u načinu rada brojača. Podaci koji se upisuju u komandni registar su selekcioni signali za multiplekser i signal dozvole rada brojača. Upisani podaci u brojač predstavljaju vremenski interval tj. gornju granicu vremena izvršenja zadatka. Brojač broji unazad, od zadate vrednosti do nule. Ako se u tom zadatom intervalu zadatak ne izvrši, na izlazu brojača TC javlja se signal koji preko interapt kontrolera šalje mikroprocesoru zahtev za prekid.

Pošto se koristi osmoulazni interapt kontroler moguće je ovim hardverskim modulom vršiti nadzor do osam procesa koji se istovremeno izvršavaju. Ako je bar jedan od ulaza u interapt kontroleru aktivan, preko INTR izlaza, generiše se zahtev za prekid tj. interrupt.

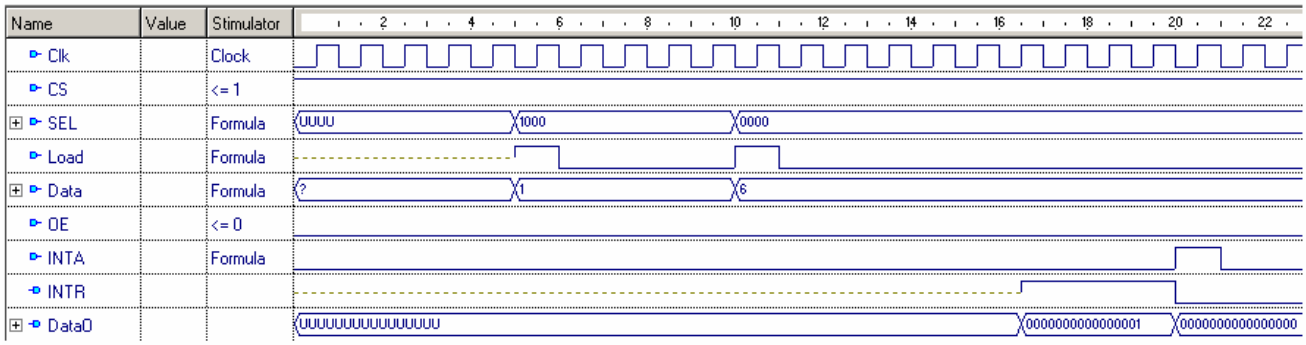
Predloženi hradverski modul opisan je jezikom VHDL, Aldec Active-HDL version 5.1 [4], [5] a zatim je i uspešno izvršena njegova simulacija. Na slici 3. prikazan je završni korak opisa gde se ceo modul može videti podeljen na dva podmodula. Prvi predstavlja svih osam brojača i pratećih elemenata, a drugi interapt kontroler.

Na slici 4. i 5. prikazani su karakteristični signali tj. rezultat simulacije predloženog modula za nadzor.

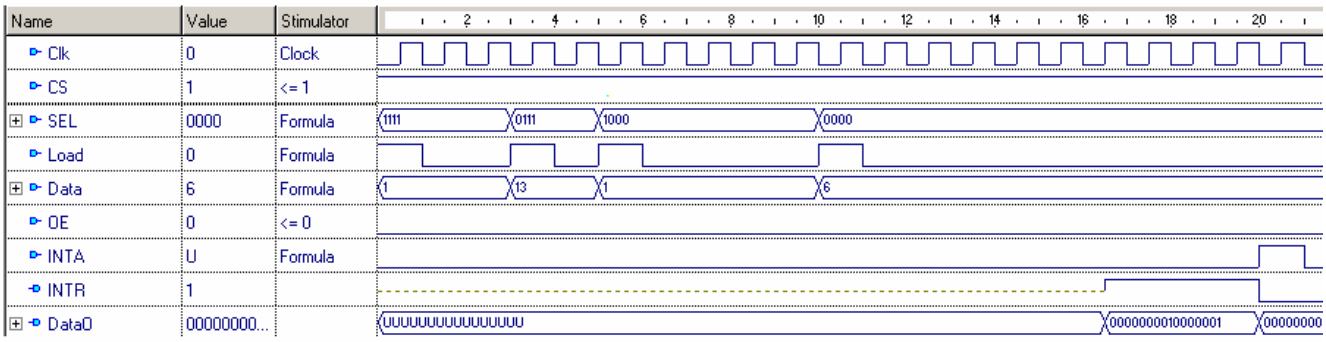
Konkretno na slici 4. upis u nadzorni modul je izvršen u trenutku t=5 kada se upisuju podaci u komandni registar (u registar se upisuje vrednost 001 što znači da EN treba postaviti na 1 a selekcioni signali za multiplekser su oblika00 tj. maksimalno vreme za izvršenje zadatka je postavljeno na 65.5ms) i u trenutku t=10 kada se upisuju podaci u brojač (u brojač se upisuje vrednost 6).



Sl. 3. *Opis predloženog rešenja pomoću blok diagram editora VHDL-a*



Sl. 4. *Rezultati simulacije-pristup nadzornom modulu*



Sl. 5. *Rezultati simulacije-pojava dva interapt zahteva istovremeno*

Signal za selekciju čipa SC je postavljen na vrednost 1 što znači da je selektovan modul za nadzor. SEL ulazi su selekcioni ulazi u dekoder na osnovu kojih se propuštaju signali za upis tj. čitanje do određenog komandnog registra tj. brojača. Pretpostavimo da se događaj nije izvršio na vreme tako da se u trenutku t=16 javlja zahtev za interapt tj. signal INTR se setuje. Nakon postavljanja signala INTA u 1, što znači da je zahtev za interapt uspešno prosleđen procesoru, INTR se resetuje i spremno čeka pojavu novog interapta. Zajedno sa setovanjem signala INTR, procesoru se šalju podaci o tome na kom ulazu se javio interapt tako da DataO ima vrednost 0000000000000001. Jedinica na datoj bitskoj poziciji znači da se interapt javio na ulazu IRQ0 interapt kontrolera.

Na slici 5. prikazan je slučaj kada se u isto vreme jave dva interapta i to na ulazima IRQ0 i IRQ7. Prvo se u trenutku t=0 vrši upis u osmi komandni registar a u trenutku t=3 u osmi brojač (u osmi brojač se upisuje 13), a zatim se, u trenutku t=5 vrši upis u prvi komandni registar i u trenutku t=10 u prvi brojač (u prvi brojač se upisuje 6). I za ovaj slučaj smo pretpostavili da se događaji nisu izvršili na vreme tako da se, istovremeno, u trenutku t=16, javljaju dva zahteva za interapt. DataO ima vrednost 000000001000001, a jedinice na prvoj i osmoj poziciji odgovaraju ulazima interapt kontrolera IRQ0 i IRQ7 gde su se javili zahtevi za interapt.

4. ZAKLJUČAK

U radu je predloženo jedno rešenje za nadzor real-time procesa koje omogućava nadzor vremenskih karakteristika za vreme debagovanja i testiranja sistema u samom procesu projektovanja kao i kontinualno on-line proveravanje svih vremenskih karakteristika u toku rada. Radi njegove realizacije modul je opisan u jeziku VHDL i prikazani su rezultati njegove simulacije.

LITERATURA

[1] N. Nissanke, *Realtime Systems*, Prentice Hall, 1997.

[2] Milun Jevtić, Branimir Đorđević, Marija Blagojević, "Jedna realizacija nadzora procesa u sistemu za rad u realnom vremenu", YU iNFO 2000, zbornik radova, mart 2000, Kopaonik.

[3] Milun Jevtić, Milunka Damnjanović and Vladimir Živković, "A Solution to Run-time Monitoring of Rreal-Time Systems", Proceedings of First Conference on Electrical Engineering & Electronics, December 1998. Gabrovo, pp. 272-277.

[4] K.C. Chang, *Digital Systems design with VHDL and Synthesis,* IEEE Computer Society, 1999.

[5] M. Damnjanović i ostali, *Praktikum laboratorijskih vežbanja iz projektovanja i testiranja elektronskih kola i sistema*, Elektronski fakultet u Nišu, 2000.

Abstract – This paper describes a solution for run-time monitoring of real-time systems realized in VHDL. This modul can be used for monitoring the timing characteristics during the debugging and testing in process design of RTS as well as for contionous on-line cheking for correct process-timing runtime in use.

MODULE FOR RUN-TIME MONITORING OF REAL TIME PROCESSES REALIZED IN VHDL

Sandra Brankov, Milun Jevtić