



**Katedra za računarsku tehniku i informatiku
Elektrotehnički fakultet
Univerzitet u Beogradu**

Master rad: Automatsko generisanje grafičkih simulatora

**Mentori
prof dr Jovan Đorđević
doc dr Boško Nikolić**

**Autor
dipl.Ing. Ivan Pantić**

Beograd, 2009

Sadržaj

Uvod.....	3
Motivacija.....	3
Cilj i opseg sistema.....	3
Postojeća rešenja	5
Kompozicija sistema	6
Osnovne funkcionalnosti sistema	7
Spoljašnja apstrakcija komponenata	7
Vrste gradivnih komponenti.....	8
Funkcionalnosti dizajnera	9
Implementacija.....	10
Model simulacije.....	10
Struktura modela simulacije.....	10
Aplikativni interfejs za kreiranje modela	12
Dizajner simulacije	13
Sinteza i interpretiranje	16
Mesta za nadogradnja	17
Zaključak	18
Reference	19

Uvod

Motivacija

Tokom dugog niza godina na Elektrotehničkom fakultetu Univerziteta u Beogradu postoji potreba za realizacijom hardverskih simulatora koji imaju pre svega edukativnu svrhu i čija osnovna namena jeste vizuelno prikazivanje i način rada različitih hardverskih komponenti. Realizacija ovakvih simulatora se obično izvodi ručno odnosno pisanjem koda u ciljnom programskom jeziku. Ovakav pristup, za manu ima veoma dugotrajan proces koji podrazumeva pisanje programa i dizajniranje grafičke reprezentacije komponenti, kao i realizovanje kompletnog grafičkog okruženja za korisnike, koji može biti veoma složen.

Prilikom dizajniranja i implementacije delova ovakvih sistema može se primetiti velika repetitivnost, odnosno pojava da su neki delovi novog sistema već realizovani u nekom od prethodnih projekata ali se ta realizacija veoma često ne može primeniti iz razloga različitih nekompatibilnosti, pre svega interfejsa tih komponenti koji se po pravilu razlikuju samo u nekim detaljima.

Takođe je uočeno, da se troši nedopustivo puno vremena na delove sistema koji mogu biti potpuno, ili u velikoj meri automatizovani, čime bi se značajno smanjilo vreme proizvodnje ovakvih simulatora i količina resursa potrebna za realizaciju.

Iz ovih razloga uočena je potreba za realizacijom alata koji bi omogućio brzu, lagodnu i pre svega efikasnu sintezu ove vrste simulatora.

Cilj i opseg sistema

Glavni cilj realizacije SAGS-a jeste razvoj alata za sintezu simulatora, i u ovom radu su predstavljeni osnovni zahtevi koje smo uočili i definisali. Cilj razvoja SAGS-a jeste realizacija softverskog alata koji omogućuje grafičko dizajniranje i automatsko generisanje svih vrsta pokaznih simulatora.

Pod pojmom pokaznih simulatora podrazumevamo svaki softverski program koji za cilj ima da na jednostavan i korisniku prihvatljiv način prikaže strukturu i funkcionisanje nekog složenog sistema. Realizacija simulatora je česta praksa za sisteme čija kompleksnost prelazi granice ljudskog razumevanja, zapažanja i pamćenja. Simulatori za cilj imaju da na postepen i cesto vizuelan način korisnika uvedu u način funkcionisanja nekog sistema. Primena je široko rasprostranjena i pokriva gotovo sve oblasti inženjerije ali se često može koristiti i u oblastima koje nisu usko vezane sa inženjerijom. Veoma često se simulatori koriste u arhitekturi, građevini, mašinstvu, elektrotehnici i drugim inženjerskim branšama ali nije zanemarljiva njihova uloga nu u predstavljanju matematičkih, medicinskih, ekonomskih i drugih modela.

Realizacija ovakvih simulatora često predstavlja dug i mukotrpan posao koji iziskuje dosta vremena, novca i drugih resursa. Drugi veliki nedostatak tradicionalnog načina razvoja jeste činjenica da se ovi simulatori najčešće realizuju za samo ciljni domen, odnosno za određenu vrlo usku delatnost što onemogućuje njihovu ponovnu upotrebu za realizaciju u drugim domenima.

Cilj SAGS-a jeste prevazilaženje ovih nedostataka. Upotrebom SAGS-a projektantu se omogućuje da grafički dizajnira ove simulatore koristeći već postojeće module kao i da dizajnira nove module sa specifičnom upotrebom vrednošću, na osnovu kojih će alat sam i potpuno automatski kreirati sve potrebne elemente i obezbediti valjano funkcionisanje. Ovime se predviđa značajno skraćanje vremena realizacije čime se smanjuje potrošnja i ostalih resursa.

SAGS ima za cilj da prevaziđe i drugi značajan nedostatak u realizaciji tako što bi se izgradnja potpuno generalizovala sa razlikom da bi različiti domeni imali sebi specifične module koji oslikavaju specifičnosti u ponašanju. Ovime se postiže da se na potpuno uniforman način mogu izgraditi vrlo složene simulacije za različite oblasti. Tako simulator koji predstavlja dizajn i način rada modernog mikroprocesora može u velikoj meri da se poistoveti sa izgradnjom simulatora koji prikazuje trenutnu produktivnost proizvodnih linija u nekoj fabrici ili simulatora koji prikazuje tok akcija koje treba preduzeti prilikom hirurške operacije.

Osnovni cilj SAGS-a jeste značajno pojednostavljenje procesa projektovanja i realizacije simulatora kao i njihova ponovna upotreba u različitim ciljnim oblastima, odnosno domenima.

Svi simulatori koji za cilj imaju predstavljanje strukture, i logike funkcionisanja nekog sistema, pri čemu se oni sastoje iz drugih podсистema i elemenata strukture koji imaju jasnu funkciju mogu biti generisani korišćenjem SAGS-a. Pod pojmom jasna funkcija se podrazumeva da komponente simulatora imaju svoje ulazne i izlazne vrednosti i logiku komponente koja zapravo predstavlja funkciju preslikavanja ulaznih vrednosti u izlazne. Primer ovakvih komponenti je recimo, hardverski sabirač koji na osnovu svojih ulaza izračunava zbir sabirajući ulazne vrednosti ili recimo preduzeće koje na osnovu ulaznih sirovina izbacuje gotove proizvode.

Treba napomenuti da osnovni cilj SAGS-a nije realizacija animacija u određenim sektorima već aktivnih simulacija. Takođe, generisani simulatori imaju namenu samo da jasno prikažu stanja u kojima se sistem nalazi tako da postoji potreba za baznim sistemom koji bi dostavljao informacije ciljnom simulatoru.

Postojeća rešenja

Osmisliti laboratorijske vežbe iz određene oblasti računarskih nauka nije uvek jednostavan zadatak. Kao efikasnu pomoć savremena nastava koristi softverske sisteme određenih karakteristika. Takav slučaj je i sa nastavom i laboratorijskim vežbama iz oblasti arhitekture i organizacije računara. Već duži niz godina, prateći veliki razvoj hardverske i softverske industrije, softverska simulaciona okruženja se koriste u ovoj oblasti na skoro svim poznatijim svetskim fakultetima i univerzitetima.

Zato su i autori ovog rada, na početku stvaranja laboratorije iz arhitekture i organizacija računara na Elektrotehničkom fakultetu u Beogradu izvršili analizu simulatora dostupnih u otvorenoj literaturi. Osnovna ideja je bila da se proverí da li postoji jedinstveni softverski sistem koji bi se koristio u laboratoriji tokom svih kurseva iz ove oblasti na više godina studija. Samim tim bi se povećala efikasnost nastave, jer bi studenti na višim godinama već bili upoznati sa funkcijama sistema.

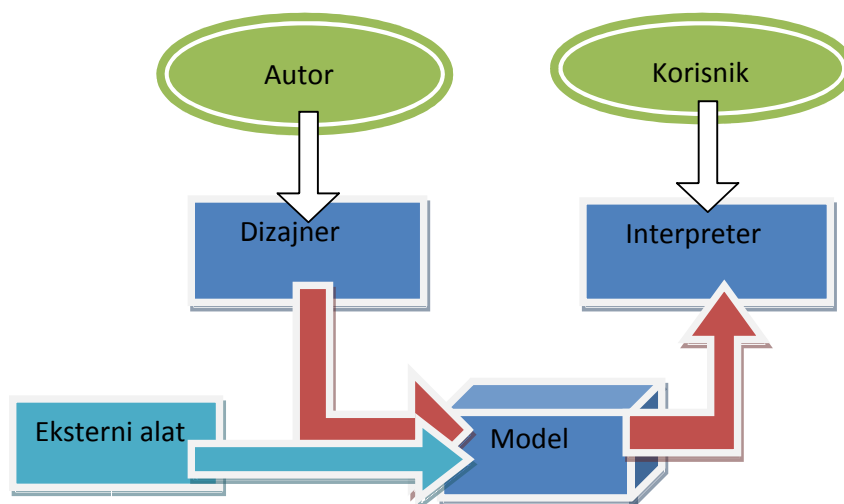
Laboratorijske vežbe obuhvataju različite koncepte i predznanje. Početne treba da demonstriraju karakteristike i ponašanje osnovnih komponenti savremenih računarskih sistema. Softverski sistem bi omogućio razvoj i simulaciju jednostavnijih komponenti. U okviru kasnijih kurseva trebalo bi raditi sa kompleksnijim tehnikama i konceptima. Studenti bi u okviru softverskog sistema imali na raspolaganju ranije dizajnirane složene delove računarskih sistema sa određenim osobinama. Postojala bi mogućnost simulacije i analize sistema pokretanjem različitih test programa, čime bi se proučavala predložena primenjena rešenja. Na kraju studenti bi koristili gotove alate da implementiraju sopstvene sisteme, a softverski sistem bi trebalo da omogući podršku za razvoj i simulaciju i najsloženijih računarskih sistema.

Postavlja se pitanje da li postoji takav softverski sistem koji bi se na opisani način koristio. Na osnovu detaljne analize izdvojeni su najprihvatljiviji sistemi, kao što su SimpleCPU, ESCAPE, DLXview, M5, JHDL i HASE. I sami autori su ranijih godina pokušavali da realizuju simulatore koje bi koristili za određene kurseve iz ove oblasti. Svi ovi simulatori su imali fiksne računarske sisteme i proučavali su određeni skup problema – osnovni rad procesora i povezivanje sa ostalim komponentama, izvršavanje kompleksnih instrukcija, pipeline organizacija procesora, organizacija i rad sa memorijskim modulima. Studenti su na različitim kursovima koristili različite simulatore, tako da su pre svakog kursa morali da se upoznaju i sa njegovim osnovnim funkcijama.

Na osnovu sprovedene analize, autori su, ipak, odlučili da treba da se realizuje novi softverski sistem koji bi studentima omogućio i sintezu računarskih sistema različitih složenosti i analizu računarskih sistema sa različitim osobinama. Tako je razvijen novi sistem čiji su opis i sama implementacija dati u narednim odeljcima.

Kompozicija sistema

Kompletan sistem se funkcionalno može dekomponovati u tri celine. Centralni deo sistema uvek predstavlja model simulatora koji kao kompleksna struktura sadrži sve informacije o tekućoj simulaciji, njenim gradivnim delovima, vezama kao i o trenutnom stanju simulacije. Drugi deo je interpreter čija je uloga da intepretira aktivnu simulaciju i na adekvatan način prikazuje krajnjem korisniku njenu strukturu, tokove signala i informacije o trenutnom stanju simulacije. Poslednji deo sistema je dizajner koji predstavlja alat koji je optimizovan za dizajniranje simulacije od strane autora, izgradnju svih neophodnih gradivnih elemenata kao i definisanje ponasanja. Jednom kreiran model simulacije kreiran uz pomoć adekvatnog dizajnera se može kasnije neograničeno koristiti od strane interpretera i predstavlja krajnji produkt u procesu dizajniranja. Šema kompozicije celokupnog sistema je data na slici.



Slika 1. Komponente sistema

Kao što je na slici i predstavljeno model simulacije može biti i produkt drugih alata osim matičnog dizajnera. To znači da bi se validan model mogao dobiti i kao izlaz iz mnogih alata koji su već u upotrebi u kombinaciji sa odgovarajućim dodatkom koji bi izvršio neophodnu konverziju. Time bi se moglo uštedeti na pravljenju komponenata simulacije koji su već realizovani korišćenjem drugih alata čija topologija i logika rada može biti konvertovana u SAGS model.

U ovom radu će biti obrađen levi deo sistema sa slike što podrazumeva statički deo modela simulacije, njegovu ulogu, dizajn i strukturu kao i izgradnju dizajnera za kreiranje simulacionih modela od strane autora simulacije. Prvo će biti reči o zahtevima sa tačke gledišta autora simulacije dok će kasnije biti date i odrednice na koji način je sistem dizajniran i implementiran.

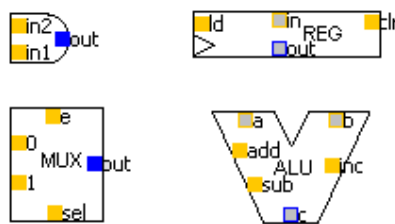
Osnovne funkcionalnosti sistema

Sistem u svojoj osnovi nudi korisniku mogućnost da koristeći već ugrađene gradivne komponente kreira komponente veće složenosti i proširene funkcionalnosti kako bi najzad dosegao nivo funkcionalnosti simulatora koji želi realizovati. Takođe autor simulacije ima mogućnost definisanja svojih gradivnih komponenata kao bi povećao nivo ponovne upotrebljivosti i tako postepeno dekomponovao veće celine na manje delove.

Spoljašnja apstrakcija komponenata

Zajednička osobina svih vrsta gradivnih komponenata je postojanje definicije njihovog spoljašnjeg izgleda. Spoljašnji izgled komponente predstavlja apstrakciju te komponente u sistemu i ukazuje korisniku na opseg funkcionalnosti koju tom komponentom može da postigne. Pri tom treba napomenuti da nivo apstrakcije u ovom slučaju može biti proizvoljan. Drugim rečima gradivna komponenta može biti najprostije logičko kolo kao što su AND ili OR ali i komponente potencijalno veoma velike složenosti kao što su ALU jedinice, moduli za obradu prekida ili sam procesor kao takav.

Pored grafičkog izgleda komponente spoljašnji izgled u potpunosti određuje i interfejs te komponente koji je neophodan pri interakciji sa drugim gradivnim komponentama. Na ovaj način gradivna komponenta postaje crna kutija koja je jasno definisana svojim reprezentativnim izgledom i interfejom koji zadovoljava bez obzira koji nivo kompleksnosti nosi u sebi. Interfejs komponente je predstavljen skupom pinova preko kojih neka komponenta može da prima informacije iz spoljašnjeg sveta i da ih generiše na svojim izlazima. Može se primetiti da se lako prepoznaje potreba za postojanjem skupa ulaznih pinova (I) i skupa izlaznih pinova (O) svake komponente što je i ugrađeno u sistem. Na ulaznim pinovima komponenta osluškuje dešavanja u spoljem svetu i na osnovu njih generiše vrednosti na svojim izlaznim pinovima. Primeri nekih komponenata su dati na slici.



Slika 2: Spoljni izgled komponenata u dizajneru

Treba primetiti da ovde nije napravljena distinkcija između kombinacionih i sekvencijalnih mreža jer za time nema potreba s obzirom na činjenicu da obe vrste mreža imaju isti

interfejs i da je jedina razlika u funkciji prelaza (funkcija koja generiše vrednost izlaznih pinova). Dok je kod kombinacionih mreža ova funkcija potpuno direktna:

$$O = F(I)$$

kod sekvencijalnih mreža ona dodatno zavisi i od prethodnog stanja mreže:

$$\begin{aligned} O &= F(I, S_{\text{old}}) \\ S_{\text{new}} &= G(S_{\text{old}}) \end{aligned}$$

Gde S predstavlja stanje mreže i uvek se ažurira u novo stanje preko funkcije G.

Bez obzira na ovu razliku funkcija prelaza treba da bude enkapsulirano znanje samog modula i da sa stanovišta korišćenja ovih komponenti nema nikakve funkcionalne razlike. Jedina razlika koja se može primetiti kod ovih komponenta je postojanje dodatnih signala takta koji donose informaciju komponenti kada funkcija prelaza treba da se okine i kada vrednosti na izlaznim pinovima treba da budu ažurirane.

Vrste gradivnih komponenti

Sistem razlikuje dve osnovne vrste gradivnih komponenti a to su: elementi i moduli.

Element je gradivna komponenta koja nema svoju unutrašnju strukturu definisanu uz pomoć drugih gradivnih komponenta već je njeno ponašanje definisano funkcijom prelaza :

$$O = F(I)$$

Ova funkcija se u dizajneru sistema može definisati čime se definiše kako će se vrednosti izlaznih pinova menjati u zavisnosti od vrste pobuda na ulaznim pinovima. Takođe, treba napomenuti da elementi predstavljaju atomske jedinice modela simulatora i da krajnji korisnici neće moći da istražuju unutrašnjost ovih komponenta već one predstavljaju crnu kutiju sa stanovišta ponašanja. Ako se neki simulator realizuje do nivoa logičkih kola onda će elementi predstavljati elementarne jedinice kao što su OR ili AND kola dok se u nekim slučajevima višeg nivoa apstrakcije elementima mogu predstavljati veoma kompleksne celine kao što su aritmetičko-logička jedinica ili pak određeni delovi samog procesora zaduženi za realizaciju određene funkcionalnosti.

Modul je vrsta gradivne komponente čija je funkcionalnost predstavljena uz pomoć drugih gradivnih komponenta međusobno povezanih u celinu. Moduli se uvek sastoje od proizvoljnog broja drugih modula i elemenata koji povezani imaju određenu logičku funkciju i mogu biti upotrebljeni kasnije za dalju izgradnju.

Kompletan sistem je realizovan sa namerom da u potpunosti podrži iterativnost u razvoju željnog simulatora. Ova iterativnost se ogleda u činjenici da se na samom početku razvoja

kreira modul kome se nadalje dodaju druge gradivne komponente u smislu proširivanja njegove funkcionalnosti.

Sam model sistema treba da podrži obe vrste komponenata kao i da čuva informaciju o njihovoj povezanosti, odnosno međusobnom odnosu i interakciji dok dizajner ima za cilj da omogućiti autorima simulacije laku manipulaciju komponentama, uvoz već postojećih i realizovanje novih komponenata.

Funkcionalnosti dizajnera

Sistem iz grafičkog dizajnera omogućava korisniku da u potpunosti definiše pomenute komponente. Obezbeđeno je kreiranje proizvoljnog broja gradivnih komponenata različitih namena koje su uređene po paketima komponenata. Takođe od proizvoljnog broja paketa se mogu kreirati biblioteke za dalju distribuciju i ponovno korišćenje u drugim projektima.

Takođe, potrebno je da dizajner ima sve osobine editora u kome se mogu efikasno menjati već postojeći modeli ili kreirati novi od početka. Sam dizajner ima u sebi ugrađene biblioteke osnovnih komponenti koji su u veoma čestoj upotrebi kada je reč o hardverskim simulacijama i koje predstavljaju elementarne činioce za izgradnju.

Implementacija

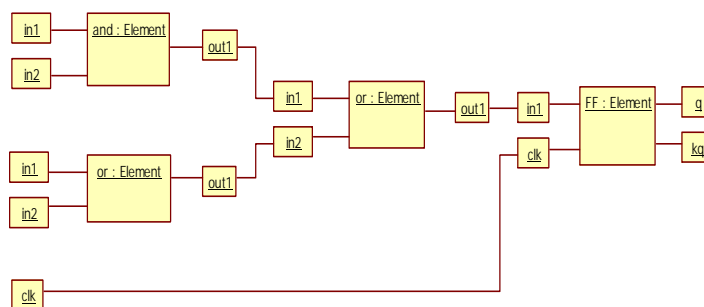
U ovom odeljku daće se osnovne smernice u kojima je softver dizajniran i implementiran. Kao što je već napomenuto u ovom radu ću se koncentrisati na razvoj dva dela sistema: samog modela simulacije i alata za dizajniranje simulacije.

Model simulacije

Centralni deo sistema predstavlja model simulatora koji je jedna kompleksna struktura hijerarhiski uređenja za čuvanje modela svih gradivnih komponenata i međusobne pripadnosti.

Sam model ima sposobnost perzistencije na disku tako što se može sačuvati u ciljne fajlove na fajl sistemu kao i iz njih nahnadno učitati.

Model simulatora predstavlja skup objekata napunjenih u memoriji koji svojim atributimai međusobnim vezama treba u potpunosti da predstavi strukturu svakog simulatora, elementa, modula,.. Primer jednog prostog modela simulacije je prikazan na slici 3.



Slika 3. Objektna reprezentacija prostog modela simulacije

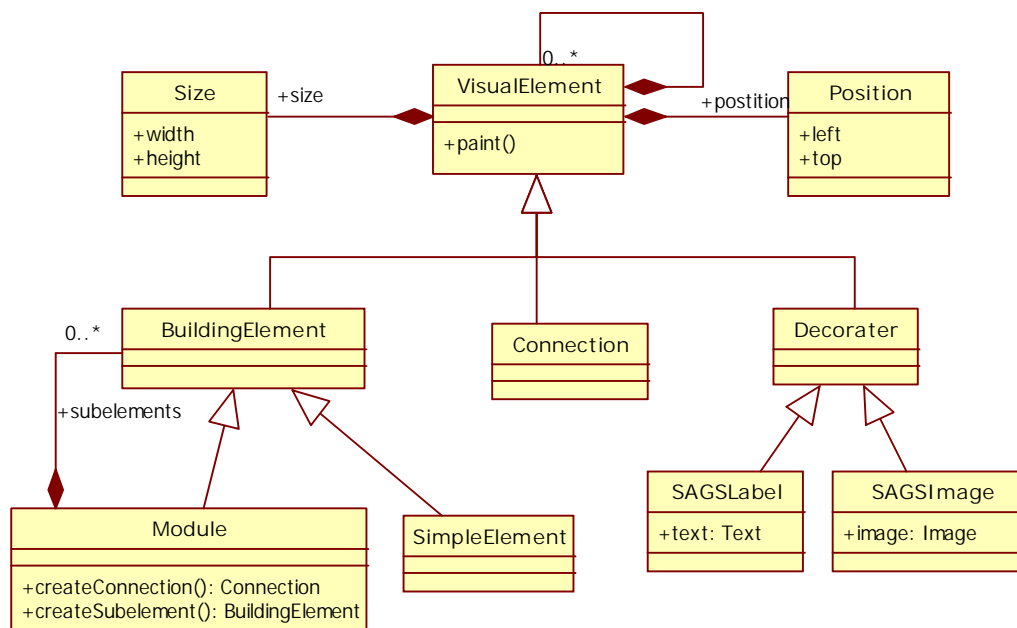
Pored statičnog dela koji reprezentuje strukturu modela model ima i svoju funkcionalnu stranu, odnosno način da u svakom trenutku model čuva informacije o trenutnom stanju simulacije. To znači da svaki pin ima svoju vrednost dok svaki prost element ima izgrađen svoj aktivni objekat koji osluškuje ulazne pinove i proizvodi odgovarajući rezultat na izlazu. Ovaj deo modela kao i interpreter koji ovo ponašanje prenosi do korisnika nisu tema su drugor rada pa stoga neće biti detaljnije obrađivana.

Struktura modela simulacije

Na slici 4. je data globalna organizacija modela simulacije. Može se primetiti da je svaki vidljivi element za korisnika simulacije predstavljen objektom VisualElement i da je uvek određen svojom veličinom na panelu simulacije i svojom pozicijom. BuildingElement je apstrakcija elemenata i modula, dok klasa Connection predstavlja apstrakciju za veze između pinova koji pripadaju komponentama.

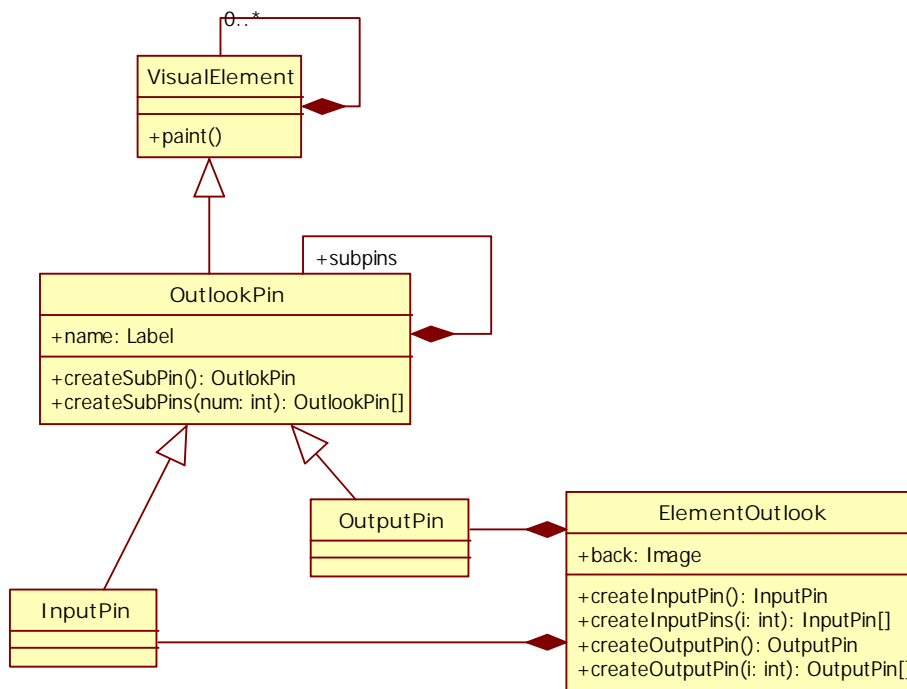
BuildingElement je apstrakcija oba tipa komponenti koji su iz nje izvedeni: modula i prostih elemenata i ovde se može primeti realizacija kompozicije između ovih elemenata, odnosno činjenice da moduli, kao složene komponente, mogu sadržati druge module ili proste elemente u svojoj gradivnoj strukturi.

Na slici se može uočiti i klasa Decorater koja je apstrakcija za proizvoljnu dekoraciju simulacione površine. Trenutno su u sistemu izgrađena dva ključna dekoratera. Prvi je zadužen za tekstualnu dekoraciju (SAGSLabel) koja se može proizvoljno formatirati i pozicionirati na radnu površinu u cilju dodatnog deskriptivnog pojašnjavanja nekih delova simulacije. Ove labela mogu sadržati uputstva za korisnike simulacije, pojašnjenje značaja nekih signala ili pak za neki drugi opis komponentata simulacije. Drugi često korišćen dekorater je grafički opis (SAGSImage) koji omogućava da se proizvoljna slika ili crtež može postaviti na simulacionu površinu a sa istim ciljem, dodatnog pojašnjavanja nekih delova.



Slika 4. Globalna organizacija sistema

Što se tiče spoljašnjeg izgleda komponentata treba napomenuti da svaki BuildingElement sadrži instancu klase ElementOutlook. Na ovom mestu su enkapsulirani svi podaci o izgledu komponente, pre svega slike koja je pozadina komponente i koja ukazuje na logičko značenje komponente a zatim i na kolekcije ulaznih, odnosno izlaznih pinova koje autor simulacije može proizvoljno definisati. Dijagram organizacije pinova je dat na slici 5.



Slika 5. Organizacija pinova

Pinovi, kao i svi signali u simulaciji, mogu biti proizvoljne širine, imaju mogućnost da budu sakriveni, odnosno nevidljivi u simulaciji kao i da im se postavi inicijalna vrednost koja će biti konzumirana kada simulacija počne.

Kompletan opisani model ima mogućnost da se eksportuje u izlazni XML fajl. Ovaj fajl sadrži reference objekata, njihove veze i reference na spoljne resurse. Spoljni resursi su slike, koje predstavljaju grafičku reprezentaciju komponenata kao i proizvoljne grafičke dekoracije. Svi ovi resursi zajedno sa XML definicijom su na kraju sačuvani u jednoj ZIP arhivi koja zapravo predstavlja format modela i ima ekstenziju .mod. Kompletan proces eksportovanja, kao i inverzni proces učitavanja je ugrađen u sam model i sastavni je deo njega. On održava sve međusobne reference i tako omogućava potpunu perzistenciju. Kao rodukt se najzad dobija jedan fajl koji predstavlja modul najviše hijerarhije u definisanoj simulaciji i koji se može učitati u interpreter simulacije.

Aplikativni interfejs za kreiranje modela

Model sadrži i definisan uprošćen API koji mogu koristiti korisnici koji eksterno žele kreirati komponente ili kompletne simulacije. Primer kreiranja jednog modula je dat u primeru 1. Ovoj aplikativni interfejs je namenjen onima koji iz Java žele kreirati simulacioni model sa jedne strane značajno uprošćen i jednostavan za korišćenje, jer automatski kreira veze između objekata i definiše određena polja, dok je sa druge strane u potpunosti funkcionalan i ne umanjuje opštost modela. Sama realizacija dizajnera je

takođe bazirana na ovom API-ju. Takođe, realizacije konevertera koji treba da izvezu strukture modela iz drugih alata potencijalno mogu da koriste ovaj interfejs.

```
Module m = BuildingElement.createModule();
ElementOutlook el = m.createOutlook();
el.setDimension(100, 100);
el.setBackgroundImage(new BufferedImage(20,20,1));
el.createInputPin("in0");
el.createInputPin("in1");
el.createOutputPin("out0");

el.createLabelDescriptor("Element outlook");

m.createSubelement(m);

BuildingElement.storeElement(m, new FileOutputStream(
"./samples/rs/sags/model/test/moduleTest1.mod"));
```

Primer 1. Korisnički kod za kreiranje modula

Dizajner simulacije

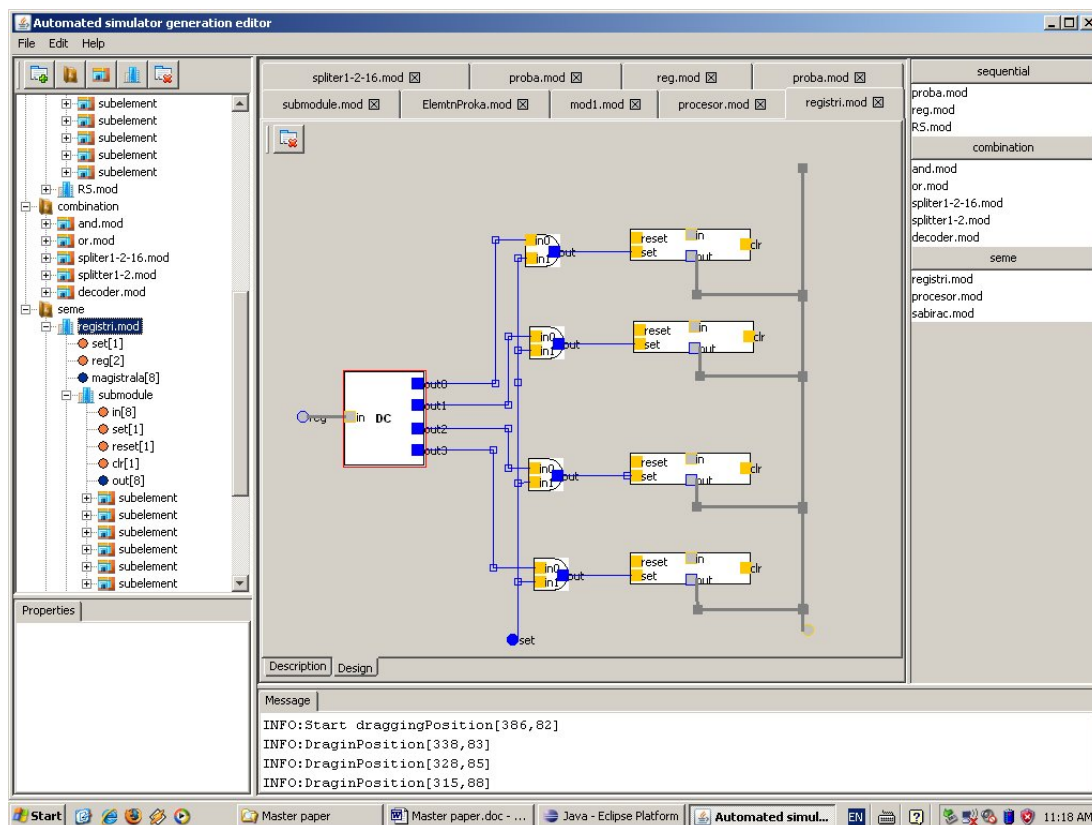
Dizajner simulacije je standalone aplikacija koja ima interfejs prilagođen korisnicima, odnosno autorima simulacije i omogućava direktne manipulacije nad modelom.

Izgled aplikacije je prikazan na slici 6. U levom delu ekrana se nalazi stablo koje predstavlja trenutni radni prostor (workspace). Na najvišem nivou hijerarhije se upravlja projektima. U okviru projekta se može definisati proizvoljan broj paketa unutar kojih se, nadalje, upravlja gradivnim elementima, bili ono prosti elementi ili moduli. U gornjem delu kontrole postoje tasteri koji reprezentuju akcije za kreiranje novog projekta, paketa, prostog elementa ili modula. Takođe komponente se mogu i brisati iz stabla i tako uklanjati iz sistema. Stablo omogućava navigaciju kroz komponente simulacije i i njega se mogu otvoriti detaljniji pogledi na strukturu komponentata.

U desnom delu ekrana nalazi se prozor sa dostupnim elementima za izgradnju (toolbox). U jednom trenutku aktivan je samo jedan projekat. Trenutno aktivan projekat je onaj čiji modul se trenutno dizajnira i aktivan je u editoru. U ovom delu liste elemenata dostupni su sve komponente unutar aktivnog projekta, zatim komponente koje se nalaze u predefinisanoj i već ugrađenoj biblioteci komponentata kao i komponente dostupne iz korisničkih biblioteka. One su grupisane po paketima iz razloga lakšeg pronalaženja i manipulacije. Klikom na neku od ovih komponenti pravi se njena kopija i korisnik je može pozicionirati na proizvoljno mesto u dizajneru čime ona odpočinje svoj život unutar te komponente i može se menjati i prilagođavati bez da se bazična komponenta promeni.

U centralnom delu se može otvoriti proizvoljan broj komponenti za dizajniranje. Svaka otvorena komponenta ima dva pogleda predstavljena u dva taba "Description" i "Design". Koristeći prvi korisnik definiše spoljašnji izgled komponente kreirajući njenu reprezentaciju i kreiranjem proizvoljnog broja ulaznih i izlaznih pinova. Takođe

obezbeđen je mehanizam zumiranja što je vrlo pogodno u slučaju minijaturnih komponenta.



Slika 6. Korisnički izgled dizajnera simulacije

Koristeći odeljak “Design” korisnik definiše unutrašnju implementaciju komponente. U zavisnosti od vrste komponente na ovom mestu će se otvoriti različiti prozori. U slučaju da je reč o prostom elementu korisnik će ovde u obliku izraza definisati funkciju prelaza ulaza izlaza dok će se u slučaju modula ovde otvoriti površina za dizajniranje na kojoj korisnik stavlja komponente izabrane iz liste alata i međusovno ih povezuje. U svakom trenutku korisnik može izbrisati neku od komponenti, kreirati ili ukloniti neku od veza ili eksplicitno kreirati magistralu koja ima posebna svojstva i stanja.

Pri kreiranju svakog ulaznog ili izlaznog pina u spoljnjem opisu komponente kreira se njegov konjugovan pin u modelu unutrašnjeg dizajna koji je sa njim usko povezan ali je potpuno nezavistan element i može se proizvoljno pozicionirati. Preko ovog pina se signal sa interfejsa komponente prenosi u njenu unutrašnjost i tako omogućava dalja povezivanja.

Treba napomenuti da je izgled same šeme simulacije ovde prilagođen potrebama koje nameće dizajner. To znači da su pinovi veoma jasno nacrtani kako bi se mogli lako

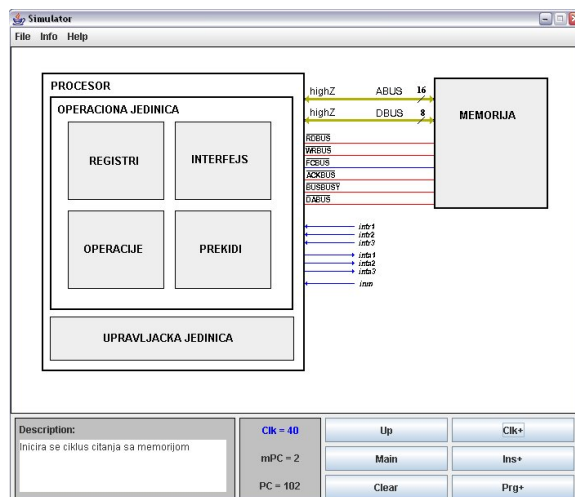
selektovati. Takođe, slično je sa tačkama na kojima se linije preklapaju kao i sa nekim drugim dekoracijama. Ovakav izgled simulacije je prilagođen i predstavlja podešen doživljaj za editovanje (look and feel). Kada se simulacija prezentuje korisnicima koristi se drugi look and feel pa je stoga izgled za korisnika nešto izmenjen, odnosno ima naglašene neke druge elemente koji su od interesa za korisnike. Ovde treba napomenuti da sistem ima nekoliko ugrađenih izgleda koje i sam korisnik, kao i autor simulacije može podešavati.

U donjem delu ekrana se uobičajeno nalazi odeljak u kome se mogu menjati svojstva selektovane komponente i zapisnik korisnikovih akcija kao informacije o potencijalnim greškama i upozorenjima kada je dizajn u pitanju.

U svakom trenutku dizajniranja na osnovu izgrađenog modela korisnik može startovati sam simulator za koji je sam model potpun izvor informacija za generisanje finkcionalnog simulatora. Pri tome se pokreće interpreter koji za ulaz ima trenutno aktivnu komponentu i koji korisniku prikazuje živu simulaciju. Sve promene koje korisnik napravi u strukturi odražavaju se isključivo na srednji sloj aplikacije, odnosno na model simulatora koje se ogledaju u kreiranju novih objekata raznih tipova, njihovim uništavanjem, povezivanjem ili uništavanjem veza između njih.

Sinteza i interpretiranje

Kao što smo već napomenuli u svakom trenutku razvoja nekog modela korisnik može da sintetiše simulator na osnovu trenutnog modela. Jedan potencijalni izgled sintetisanog simulatora je predstavljen na slici 7.



Slika 7: Izgled generisanog simulatora

U centralnom delu simulatora se nalazi sama predstava simulacije kroz koju korisnik može da se kreće po nivoima simulacije selektovanjem određene složene komponente. Međusobne veze između modula su prikazane linijama različite boje i debljine u zavisnosti od trenutnog stanja simulacije kao i od vrste veze.

U donjem delu ekrana se nalazi kontrolna tabla sa koje korisnik može da upravlja simulacijom. Treba napomenuti da realizacija samog modela simulatora i mehanizam tranzicija iz jednog stanja modela u drugu u potpunosti omogućen koncept praćenja istorije izvršavanja simulacije pa je samim tim simulaciju moguće vraćati i u nazad za predefinisani broj tranzicija odnosno taktova.

Mesta za nadogradnja

Postoji nekoliko otvorenih mesta u trenutnoj realizaciji koji bi u značajnoj meri povećali upotrebljivost opisanog sistema a ovde ćemo navesti neke.

Sam sistem preko već opisanih interfejsa u potpunosti podržava povezivanja sa drugim sistemima ali trenutno ova podrška ne postoji. Kao što smo već pomenuli sam model simulatore predstavlja potpun i dovoljan interfejs za generisanje simulatora a činjenica da interfejs "Import" obezbeđuje potpunu funkcionalnost je idealno mesto saradnju sa drugim sistemima. Preko ovog interfejsa se mogu učitavati moduli iz drugih jezika, na primer. Među potencijalnim jezicima se ovde pre svega misli na VHDL kao jezik u kome se moduli mogu u potpunosti definisati a zatim logika funkcionisanja može bit prevedena u simulator. Takođe ovo znači da bi se već konstruisane komponente u VHDL-u mogle koristiti podjednako ravnopravno sa komponentama definisanim u matičnom editoru.

Pored VHDL-a preko ovog interfejsa bi se mogao konstruisati model i iz drugih grafičkih alata koji već podržavaju rad sa grafičkim komponentama i u tom slučaju bi oni mogli biti supstitucija za matični eritor.

Sledeća stvar koja bi korisnicima mnogo olakšala posao pri definisanju ponašanja elementa ili modula je uvođenje posebnog jezika (expression language) koji bi obezbedio jednostavnu i konciznu definiciju funkcije ulaza izlaza sa jedne strane i povećao ekspresivnost samih izraza sa druge strane. Takvi izrazi bi se nadalje interpretirali od strane posebnog modula i ugrađivali u sam model simulatora.

Zaključak

Sistem SAGS predstavlja jezgro sistema koji za krajnji cilj ima realizaciju alata za automatsko generisanje grafičkih simulatora. Osnova vodilja pri realizaciji ovog alata je bila potpuna fleksibilnost u realizaciji kako bi se obezbedila laka dalja nadogradnja i implementacija novih funkcionalnosti. Tu pre svega treba uzeti u obzir realizaciju modula za saradnju sa drugim sistemima koji bi u značajnoj meri povećali upotrebljivost i iskorišćenje trenutne realizacije. Međutim i pored toga sistem je potpuno funkcionalan i zaokružen proizvod koji se može efikasno koristiti jer sadrži sve neophodne celine za samo dizajniranje modela simulacije kao i za njegovo interpretiranje krajnjim korisnicima.

U ovom radu su detaljno bili obrađeni statički deo modela simulacije i alat za dizajniranje simulacije dok su dinamički elementi modela kao i implementacija interpretera obrađeni izvan ovog rada. Oni zajedno predstavljaju jednu celinu i u pogledu implementacije finalnog produkta i u pogledu obrađenih tema kroz ove radove.

Reference

- [1] Jovan Đorđević : "Arhitektura i organizacija računara", 2006
 - [2] Erich Gamma: "Design Patterns: Elements of Reusable Object – Oriented Software", Addison-Wesley, 1998.
 - [3] Dragan Milicev: "Objektno - orijentisano modelovanje na jeziku UML", Mikro knjiga, 2001.
 - [4] Martin Fowler: "UML Distilled: A Brief Guide to the Standard Object Modeling Language", Addison – Wesley, 2004.
 - [5] David Patterson, John Hanessy: "Computer Architecture and design: The Software/Hardware Interface", Zal. Morgan Kauman Pub. Inc, 1997.
-