Jedno proširenje razvojnog integrisanog okruženja za namenske platforme

Tijana Mašić, Nenad Četić, Jelena Kovačević, Vladimir Kovačević

Sadržaj — U radu je prikazana realizacija programske podrške nazvane XML editor, za automatsko stvaranje osnovnih blokova obrade na digitalnim signal procesorima, u okviru integrisanog razvojnog okruženja CLIDE.

XML editor, automatski stvara XML datoteke koje opisuju osnovne blokove obrade grafičkog razvojnog okruženja i na taj način značajno pojedostavljuje rad sa namenskom DSP platformom.

Ključne reči — DSP, XML, editor, GUI, IDE

I. Uvod

Razvoj aplikacije na digitalnim signal procesorima podrazmeva korišćenje niza alata: editora izvornog koda, asemblera, povezivača, prevodioca, programske podrške za uklanjanje grešaka kao i razvojnog okruženja za ciljnu paltformu. Integrisano razvojno okruženje objedinjuje skup nabrojanih alata u jednu celinu, što umnogome olakšava razvoj, pokretanje, ispitivanja i kontrolisano izvršavanje DSP aplikacija.

U okviru ovog rada analizirano je integrisano razvojno okruženje CLIDE (Cirrus Logic Integrated Development Environment)[1], razvijeno za podršku Coyote 32 familije DSP procesora firme Cirrus Logic, zasnovano na proširivoj razvojnoj platformi Eclipse[2]. Procesori iz Coyote 32 familije su tridesetdvobiti DSP procesori koji rade sa aritmetikom u fiksnom zarezu i poseduju dve odvojene memorije za podatke, X i Y, i programsku memoriju[3].

Eclipse je skup projekata koji imaju za cilj stvaranje proširive platforme za razvoj i održavanje programske podrške kroz ceo njen životni ciklus.

Jedno od postojećih proširenja CLIDE razvojnog okruženja predstavlja DSP Composer[4]. DSP Composer omogućava razvoj aplikacija na vrlo apstraktnom nivou putem grafičkog povezivanja različitih blokova obrade audio signala. Osnovni blokovi obrade se prevlače na radnu površinu DSP Composer-a i spajaju provodnicima kako bi se ostvario tok audio signala. Potom se generiše izvršni kod koji se spušta na platformu sa ograničenim resursima i izvršava u realnom vremenu.

Ovaj rad je delimično finansiran od Ministarstva za nauku Republike Srbije, projekat 12004, od 2008. god.

Tijana Mašić, Fakultet tehničkih nauka u Novom Sadu, Trg Dositeja Obradovića 6, 21000 Novi Sad, Srbija,(e-mail: tijana.masic@rt-sp.com)

Nenad Četić, Fakultet tehničkih nauka u Novom Sadu, Trg Dositeja Obradovića 6, 21000 Novi Sad, Srbija, (e-mail: nenad.cetic@rt-sp.com)

Jelena Kovačević, Fakultet tehničkih nauka u Novom Sadu, Trg Dositeja Obradovi ća 6, 21000 Novi Sad, Srbija,(e-mail: jelena.kovacevic@rt-sp.com)

Vladimir Kovačević Fakultet tehničkih nauka u Novom Sadu, Trg Dositeja Obradovi ća 6, 21000 Novi Sad, Srbija,(e-mail: vladimir.kovacevic@rt-rk.com)

Osnovni blokovi obrade su definisani sa tri XML datoteke i sa jednom ili više objektnih(.O) datoteka [5]. Tri XML datoteke su:

- ***device_name.if.xml*** – opisuje spregu procesnog bloka, sadrži početne parametre uređaja i kontrole izvršavanja
- ***device_name.pres.xml*** – sadrži informacije na osnovu kojih grafičko radno okruženje DSP Composera prikazuje osnovni blok obrade
- ***device_name.imp.xml*** – sadrži informacije o strukturama podataka i kodu potrebnom da se osnovni blok obrade uključi u projekat.

Sve tri XML datoteke počinju istim nazivom, koji predstavlja jedinstvenu identifikaciju procesnog bloka.

Pri razvoju novih osnovnih blokova obrade potrebno je definisati sve tri datoteke zasebno. Ovaj postupak zahteva poznavanje XML jezika, kao i poznavanje strukture sadržaja tih XML datoteka.

Da bi se smanjilo vreme razvoja osnovnih blokova obrade i unošenja ljudske greške, potrebno je automatizovati proces stvaranja XML datoteka.

Analiza postojećih rešenja ukazala je da slična razvojna okruženja ili nisu automatizovala proces stvaranja novih osnovnih blokova obrade (kao Audio Weaver okruženje firme DSP Concepts [6]) ili kosite kompilkovane alate za automatizaciju koji zahtevaju dodatnu obuku korisnika (primer je PurePath Studio firme Texas Instruments-a [7]).

U ovom radu je predložena programska podrška nazvana XML editor koja obezbeđuje automatizaciju stvaranja XML datoteka na jednostavan način bez prethodne obuke korisnika.

II. Analiza problema

Sadržaj koji treba da se nalazi u svim XML datotekama poseduje određenu formu koju treba poštovati prilikom stavaranja svakog novog osnovnog bloka obrade.

Unutar sve tri XML datoteke koriste se XML prostori imena:

- <*xpp:*> – označava elemente koje će XML predprocesor da obradi u DSP Composeru
- <*xpr:*> – označava elemente ili atribute unutar kojih se nalaze algebraski izrazi čija vredost treba da se izračuna
- <*sxpr:*> – označava aritmetičke izraze čija vrednost treba da se izračuna tokom XML obrade

XML predprocesor može da razlikuje nekoliko sintaksnih konstrukcija, a najčešće su prikazane na Sl.1. Konstrukcija <*xpp:include*> semantički je ekvivalentna konstrukciji #include kod C predprocesora. <*xpp:define*>

konstrukcija definiše skup elemenata i povezuje ih sa <*definition name*>. Zatim <*xpp:insert*> konstrukcija predstavlja referencu na <*xpp:define*> konstrukciju. Elementi <*xpp:define*> konstrukcije se u XML-u ubacuju na mesto pojavljivanja <*xpp:insert*> konstrukcije. <*xpp:repeat*> konstrukcija predstavlja petlju koja se ponavlja (last – first + 1) puta. <*xpp:evaluate*> konstrukcija izračunava algebraski izraz. Najčešće se koristi za postavljanje promenjive na izračunatu vrednost.

```
<xpp:include file="name of file">

<xpp:define name="definition name">

<xpp:insert name="definition name">

<xpp:repeat index="variable"
  first="expression" last="expression">

<xpp:evaluate expression="expression">
```

Sl. 1. Najčešće korišćene sintaksne konstrukcije koje XML predprocesor razlikuje

A. Prezentaciona datoteka(device_name.pres.xml)

Definiše grafičko predstavljanje osnovnog bloka obrade, uključujući ulazno/izlazne tipove priključaka, broj priključaka, mesto smeštanja priključaka na osnovnom bloku obrade, boju osnovnog bloka obrade.

Prezentaciona datoteka sadži identifikator tipa osnovnog bloka obrade <*device_type_identifier*> , prikazan na Sl.2., koji sadrži jedinstveno ime i informaciju o verziji osnovnog bloka obrade, kao i podelemente koji opisuju familiju platformi sa ograničenim resursima za koju je razvijen osnovni blok obrade.

```
<device_type_identifier>
  <family value="coyote_dsp"/>
  <type value="abs"/>
  <version major="1" minor="1" subminor="0"/>
</device_type_identifier>
```

Sl. 2. Identifikator tipa procesnog bloka

<*xpp:define name="control_panel"*> deo predstavlja kontrolnu tablu procesnog bloka. Pozicija kontrola na kontrolnoj tabli određena je pomoću reda <row>, i kolone <column>. Kontrola omogućava upravljanje jednom ili više promenljivih osnovnog bloka obrade, koja se nalazi među javnim podacima.

<*xpp:define name="device_layout"*> deo definiše raspored i broj audio i kontrolnih priključaka procesnog bloka.

B. Implementaciona datoteka(device_name.imp.xml)

Impelemntaciona datoteka definiše informacije potrebne za stvaranje objektnog koda i struktura podataka potrebnih da se projektuje osnovni blok obrade koji se može spustiti na platformu sa ograničenim resursima.

Identifikator tipa osnovnog bloka obrade je identičan delu u prezentacionoj datoteci.

Deo o unapred određenim informacijama o osnovnom bloku obrade sadrži informacije o:

- potrošnji MIPS-a (engl. million instructions per second). Primer potrošnje MIPS-a prikazan je na Sl.3.
- adresi osnovnog bloka obrade u tabeli poziva modula
- objektnim datotekama

```
<MIPS value="((sample_rate/block_size)*
  (20+(2*block_size)))/10000000.0"/>
```

Sl. 3. Primer računanja potrošnje MIPSa

<*poke_crunch_function*> deo sadrži Pyton skripte koje će se izvršavati svaki put kada kontrola promeni vrednost

<*peek_crunch_function*> deo sadrži Pyton skripte koje će se pozivati svaki put kada kontrola treba da očita neku vrednost sa platforme sa ograničenim resursima

C. Datoteka programske sprege(device_name.if.xml)

Ova datoteka definiše ime osnovnog bloka obrade u grafičkom okruženju, koji su mu početni parametri i koje priključke poseduje. Informacije o početnim parametrima i priključcima upotpunjuju informacije iz prezentacione XML datoteke. U toj datoteci je opisan grafički izgled početnog parametra, a u datoteci programske sprege koje vrednosti on može da poseduje. Dok za priključke prezentaciona datoteka opisuje njihov izgled, datoteka programske sprege opisuje koju vrstu podataka oni prihvataju, dodeljuje im imena i jedinstvene identifikacije.

Identifikator tipa osnovnog bloka obrade je identičan delu u prezentacionoj datoteci.

<*device_property_schema*> deo sadrži definicije početnih parametara.

<*device_data_schema*> deo sadrži informacije o ponašanju kontrola tokom izvršavanja obrade osnovnog bloka obrade, i informacije o priključcima. Na priključke je moguće priključiti audio i kontrolne provodnike za prenos podataka.

D. Kod obrade procesnog bloka

Algoritam koda se izvršava u okruženju koje stvara mikrojezgro DSP Composera. Mikrojezgro sadži jednostavan operativni sistem (OS) čija je glavna uloga da bude raspoređivač za određeni broj osnovnih blokova obrade. Drugim rečima, OS-a predstavlja monitorsku petlju koja poziva rutine odgovarajućih blokova obrade po unapred definisanom redosledu. Svaki algoritam poseduje tri rutine:

- *inicijalizaciona rutina* – poziva se samo jednom, prilikom ponovnog pokretanja mikrojezgra
- *blok rutina* – poziva se svaki put kad se skupi količina podataka jednaka veličini bloka za obradu
- *pozadinska rutina* – poziva se od strane mikrojezgra svaki put kad nije zaposlen

Algoritam može da poseduje i javne i zaštićene podatke. Javnim podacima je moguće pristupiti putem matičnog mikro upravljača ili drugih porcesa koji se nalaze izvan platforme sa ograničenim resursima. Zaštićeni podaci se

koriste samo u kodu alogritma koji se izršava na platformi sa ograničenim resursima.

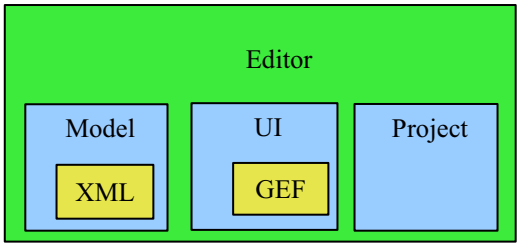
Kod algoritma se nalazi uglavnom unutar jedne izvorne (.a) datoteke, i prevodi u jednu objekntu (.O) datoteku. Imena objektnih datoteka se naznačavaju u implementacionoj XML datoteci.

Prilikom dodavanja više istih osnovnih blokova obrade na radnu površinu DSP Composera, koristi se zajednički izvorni kod, dok se memorija za podatke zauzima za svaki procesni blok posebno.

III. Opis Realizacije

Razvojno okruženje se zasniva na *Eclipse* platformi, koje je prošireno putem komponenti (eng. *plugin*).

Izvorni kod rešenja XML editora organizovan je u pakete, što je prikazano na Sl. 4.. Paketi služe za grupisanje klasa koje imaju slične funkcionalnosti, radi preglednosti koda. Ti paketi su:



Sl. 4. Prikaz grupisanja paketa

- *editor* – sadrži aktiviranje priključka
- *model* – sadrži klase koje predstavljaju elemente koji se upisuju u rezultujuće XML datoteke
- *ui* – sadrži klase za prikaz korisničkog okruženja za unos elemenata koji se upisuju u rezultujuće XML datoteke
- *project* - sadrži podatke o projektu koji se pravi
- *gef* – sadrži klase pomoću kojih se prikazuju i dodaju kontrole osnovnih blokova obrade

XML editor je realizovan kao višestranični editor [8]. Radi lakšeg unosa podataka, sadržaj svake stranice predstavlja logičku celinu. Prikaz unosa podataka u višestranični editor i upis tih podataka u XML datoteke prikazan je na Sl. 5.

A. Stranica sa osnovnim podatcima

Sadrže naziv osnovnog bloka obrade, priključke, i početne parametre osnovnih blokova obrade. Omogućeno je dodavanje novih priključaka i početnih parametara pomoću dijaloga, u kome se popunjavaju potrebni podaci. Prikaz unetih elemenata se vrši pomoću tabela .

B. Stranica sa kontrolama

Sadrže prostor za grafičko raspoređivanje kontrola osnovnih blokova obrade i dodavanje skrivenih kontrola. Kontrole se raspoređuju u željeni red i kolonu u kojoj će biti prikazani u DSP Composeru. Skrivene kotrole se dodaju pomoću dijaloga, a prikazuju pomoću odgovarajuće tabele.

C. Stranica sa podatcima o realizaciji

Sadrži podatke o objektnim datotekama, funkcijama za upis i čitanje, tipu osnovnog bloka obrade i potrošnji MIPS-a. Objektne datoteke se dodaju pomoću dijaloga, a prikazuju pomoću odgovarajuće tabele. Tip osnovnog bloka obrade se bira od ponuđenih u padajućoj listi. Omogućen je izbor datoteka koje sadrže Pyton skripte za čitanje i upis.

D. Stranica sa javnim podacima nad kojima je dozvoljeno i čitanje i pisanje

Opisuje javne podatke osnovnih blokova obrade nad kojima je dozvoljeno i čitanje i pisanje koji se nalaze u Y memoriji procesora sa ograničenim resursima.

E. Stranica sa javnim podacima nad kojima je dozvoljeno samo čitanje

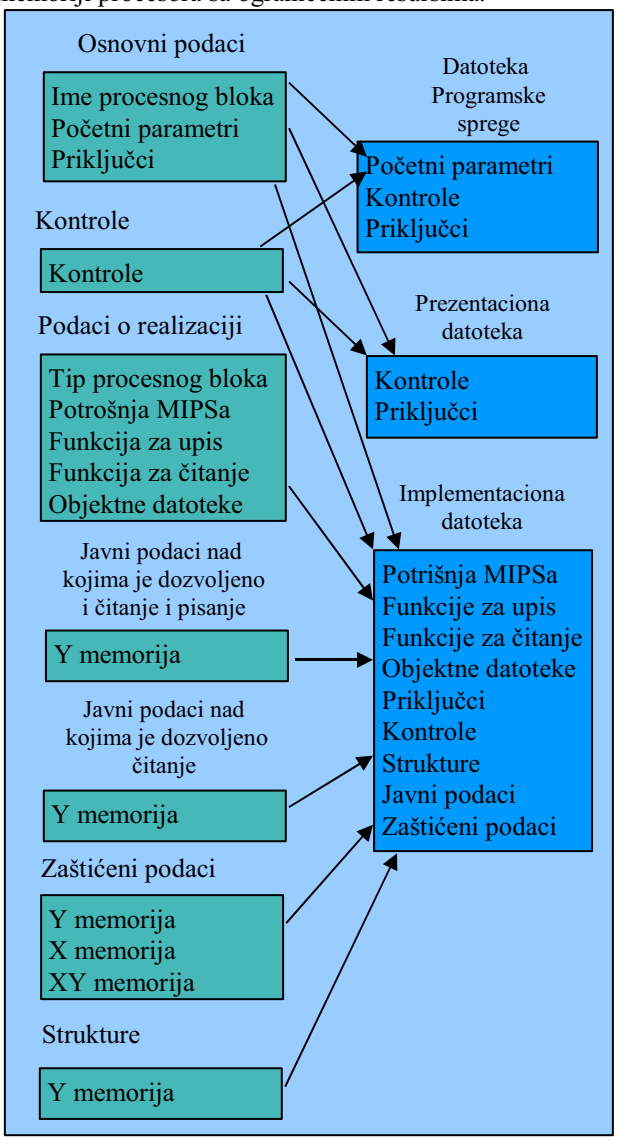
Opisuje javne podatke osnovnih blokova obrade nad kojima je dozvoljeno samo čitanje koji se nalaze u Y memoriji procesora sa ograničenim resursima.

F. Stranica sa zaštićenim podacima

Opisuje zaštićene podatke koji se nalaze u X, Y ili XY memoriji procesora sa ograničenim resursima.

G. Stranica sa strukturama

Opisuje nove strukturne tipove koje se nalaze u Y memoriji procesora sa ograničenim resursima.



Sl. 5. Prikaz unosa podataka u višestraničnom editoru, i upisa tih podataka u XML datoteke

H.Detalji realizacije

Pomoću JFace biblioteke koja je deo Eclipse projekta realizovan je prikaz, dodavanje novog sadržaja i promena sadržaja tabela[9]. JFace biblioteka obezbeđuje klase koje pomažu razvoj grafičkih elemenata koji su složeni za uvođenje. JFace omogućuje razmišljanje samo o realizaciji funkcionalnosti, oslobađajući nas brige o internoj realizaciji grafičkih objekata koji obezbeđuju željenu funkcionalnost.

Prikaz sadržaja u tabelama realizovan je pomoću prikazivača tabela (TableViewer) koji je deo JFace biblioteke. Svrha prikazivača je da pojednostave interakciju između modela koji je potrebno prikazati i grafičkog objekta koji prikazuje model. Prikazivačima je potrebno dodeliti obezbeđivač naziva, obezbeđivač sadržaja i ulazni model. Kada se prikazivaču dodeli ulazni model, on preko obezbeđivača sadržaja saznaje koje elemente iz modela je potrebno prikazati. Takođe zadatak obezbeđivača sadržaja je da obavesti prikazivač o svakoj promeni modela. Obezbeđivač naziva je zadužen za prikaz naziva elemenata modela. On prikazivaču prosleđuje tekst koji se prikazuje u tabeli.

U svakoj tabeli omogućeno je brisanje odabranih elemenata tabele, kao i pomeranje elemenata za red gore ili dole. Ovim je omogućeno korisniku raspoređivanje javnih i skrivenih podataka koji će biti upisani u memoriju platforme sa ograničenim resursima, radi povećanja efikasnosti izvršavanja algoritma na ciljanoj platformi .

Za stvaranje XML datoteka korišćen je JAXP (Java API for XML Processing) programska sprega koja omogućava raščlanjivanje XML dokumenata [10]. Korišćen je DOM(Document Object Model) raščlanjivač koji vrši predstavljanje i interakciju sa objektima unutar XML dokumenata [11]. DOM raščlanjivač stvara dokument, čiji sadržaj ima strukturu stabla, koje poseduje čvorove. Zatim se taj dokument tranformiše i upisuje u izlaznu XML datoteku.

Sl.6. Prikaz kontrola na kontrolnoj tabli

GEF predstavlja radni okvir koji obezbeđuje okruženje za grafičko uređivanje za programe koji se razvijaju na Eclipse platformi [12]. GEF pruža snažnu osnovu za stvaranje novih editora za vizuelno uređivanje korisničkih modela. Pri svakoj promeni stanja modela, GEF primećuje promene i primenjuje odgovarajuće radnje, kao što je npr. ponovno iscrtavanje modela, prilikom zahteva za pomeranjem modela. GEF je korišćen kao osnova za realizaciju rukovanja kontrolama. Pomoću njega je realizovano dodavanje novih kontrola i njihovo raspoređivanje na kontrolnoj tabli osnovnog bloka obrade, kao što je prikazano na Sl.6.

IV. Ispitivanje

Ispitivanje se obavljalo u dve faze. Prvo su 20 primitiva ispitane ručno, a zatim se na osnovu dobijenih rezlutata izvelo regresivno ispitivanje [13]. Regresivno ispitivanje (*eng. regression testing*) je vrsta ispitivanja u kome se na osnovu jednom razvijenog programa ispita više puta sprovodi ispitivanje programske podrške. Za regresivno ispitivanje korišćena je realizacija regresivnog ispitivanja DSP Composer programske podrške, razvijene na katedri za računarsku tehniku i računarske komunikacije Fakulteta tehničkih nauka u Novom Sadu[14]. Pokretanje i izvršavanje programa ispita je izvršeno upotrebom *BBT* programske podrške[15]. BBT programska podrška radi na principu crne kutije što znači da je moguće ispitivati razne sisteme bez poznavanja njihove unutrašnje strukture.

V. Zaključak

Realizacijom XML editora omogućeno je automatsko generisanje XML datoteka za opis osnovnih blokova obrade, čime je smanjeno vreme pisanja novih osnovnih blokova obrade. Takođe je smanjena mogućnost greške, jer se uglavnom pisanje novih osnovnih blokova obrade svodila na prepravljanje XML datoteka postojećih koji imaju sličnu funkcionalnost. Takođe ovo rešenje omogućuje korisnicima razvoj novih osnovnih blokova obrade bez poznavanja XML jezika.

Literatura

[1] Branislav Rankov, *Jedan pristup proširenju sistemskih programskih alata na osnovu standarda DWARF 2*, diplomski – master rad, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 2009.
[2] Steve Holzner, “Eclipse”, O'Reilly Media, 2004
[3] CS4953xx Data Sheet – Cirrus Logic
[4] DSP Composer User’s Manual – Cirrus Logic
[5] DSP Composer Primitive Elements Reference – Cirrus Logic
[6] Audio Weaver Module Developers Guide – DSP Concepts, 2004
[7]Component Publisher, Texas Instruments, Available: http://processors.wiki.ti.com/index.php/Creating_Blocks_for_C6Flo
[8] Wayne Beaton, Jeff McAffer, *Eclipse Rich Client Platform*, Eclipse Foundation, Inc, 2006
[9] Matthew Scarpino, Stephen Holder, Stanford Ng, Laurent Mihalkovic, ”SWT/JFace in Action: GUI Design with Eclipse 3.0 “
[10] James Duncan Davidson, *Java API for XML Parsing,* Sun Microsystems, Inc, 2000
[11] DOM, Available: http://www.w3.org/DOM/
[12] GEF, Available: http://eclipse.org/gef/
[13] Hiraral Agrawal, Joseph R. Horgan, Edward Krauser, Saul London, “*Incremental Regression Testing”*, IEEE Conference on Software Maintenance (ICSM ’93), Montreal, CA
[14] Ivan Stojanović, “Jedna realizacija regresivnog ispitivanja DSP Composer sistemske programske podrške na platformi sa ograničenim resursima*”, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 2010.*
[15] BB Testing, 2007 RT-SP DOO

Abstract

This paper describes implementation of XML editor that is part of integrated development environment CLIDE. XML editor automatically creates basic processing blocks for digital signal processors. XML editor automatically creates XML files for process blocks describing of grafical development enviroment, and makes process of creating them easier.

ONE EXTENSION OF INTEGRATED DEVELOPMENT ENVIRONMENT FOR EMBEDDED SYSTEMS

Tijana Masic, Nenad Cetic, Jelena Kovacevic and Vladimir Kovacevic