Embedded FPGA Linux sistemi

Dragomir El Mezeni, Milan Novaković
Mentor: dr Lazar Saranovac

***Sadržaj — Embedded* procesorski sistemi realizovani na jednom čipu, najčešće u FPGA komponentama, sa Linux operativnim sistemom pružaju veliku fleksibilnost i mogućnost brzog razvoja. Dat je pregled tipova procesorskih jezgara koji su ili implementirani ili se mogu implementirati u programabilnim komponentama, kao i najbitnije osobine Linux operativnog sistema. Prikazani su postupci i procedure potrebni za implementaciju periferija i Linux operativnog sistema na razvojoj platformi ML403 sa Virtex-4 FX FPGA čipom i PowerPC procesorom.**

***Ključne reči — Embedded* sistemi, FPGA, SoC, Linux, PowerPC**

I. UVOD

Digitalni sistemi realizovani u programabilnim FPGA (*Field-programmable gate array*) komponentama su postali veoma značajan deo današnjih zahtevnih *embedded* sistema. Povećanje logičkog kapaciteta i smanjenje cene je omogućilo ekonomski isplativu implementaciju računarskih sistema (jednog ili više) u FPGA čipu. Na tržištu postoje biblioteke gotovih delova FPGA sistema koje su optimizovane i testirane. Ove gotove komponente se nazivaju IP jezgra i nude ih proizvođači FPGA čipova ili druge firme najčešće uz odgovarajuću novčanu naknadu.

Posebna fleksibilnost *embedded* sistema se postiže implementacijom Linux operativnog sistema. Na tržištu je dostupan veliki broj gotovih besplatnih rešenja Linux operativnog sistema i uslužnih programa.

Postojanje gotovih biblioteka, korisnih razvojnih alata i veliki stepen konfigurabilnosti FPGA čipa kao i Linux operativnog sistema omogućava brz, jeftin i pouzdan razvoj *embedded* sistema.

II. FPGA SISTEMI NA ČIPU

FPGA predstavlja logičku komponentu koja se sastoji od programabilnih logičkih blokova, programabilnih ulaza-izlaza i programabilnih linija veze. Korišćenjem FPGA čipova je proces projektovanja dosta ubrzan a proces testiranja i debagovanja, znatno olakšan zbog mogućnosti jednostavnog rekonfigurisanja delova sistema u toku projektovanja. Sa porastom logičkog kapaciteta postalo je moguće implementirati procesorsko jezgro, periferije i specijalizovane delove u logici FPGA čipa. Uobičajen naziv za kompletne sisteme realizovane u jednom čipu je SoPC (*System on Programmable Chip*) ili generički SoC (*System on Chip*).

Postoje dve vrste procesorskih jezgara koje se koriste u realizaciji SoPC sistema: *hard-core* IP i *soft-core* IP.

Hard-core IP procesor predstavlja deo FPGA čipa namenjen isključivo procesorskoj logici. Realizovan je na istoj silicijumskoj osnovi i u slojevima veza povezan sa programabilnom logikom. Ovakvo procesorsko jezgro je fiksne strukture i nalazi se na tačno utvrđenom mestu na čipu. Prednosti *hard-core* procesora su te što su ova jezgra optimizovana po površini, brzini i potrošnji. Električne specifikacije ovih procesora su fiksne i nezavisne od ostatka sistema. Procesorsko jezgro je obično standardne arhitekture (PowerPC, ARM) i ima veoma dobru podršku u vidu već postojećeh kodova, biblioteka, prevodioca, debagera i operativnih sistema. Cena sistema je manja pošto je zauzeće logike svedeno na minimum. Vodeći proizvođači FPGA čipova nude čipove sa ugrađenim *hard-core* IP procesorom. Tako *Xilinx* u Virtex II PRO, Virtex-4 FX i Virtex-5 FXT ugrađuje PowerPC procesorsko jezgro, *Atmel*-ove familije *AT94K* i *AT94S FPSLIC* (*Field Programmable System Level Integrated Circuits*) imaju ugrađene *AVR* mikrokontrolere, dok *Altera* nudi *Excalibur* sistema sa ugrađenim ARM922T mikrokontrolerom. Nedostatak ovakvih procesorskih jezgara je smanjena fleksibilnost sistema usled smanjene konfigurabilnosti samog jezgra. Samim tim ne postoji mogućnost prilagođavanja jezgra konkretnoj nameni, što je veoma značajno pri projektovanju *embedded* sistema. Problem konfigurabilnosti samog jezgra može se rešiti korišćenjem *soft-core* IP procesora.

Soft-core IP procesor predstavlja mikroprocesor koji je opisan u HDL (*Hardware Description Language*) kodu i kasnije postupkom logičke sinteze implementiran na FPGA čipu. Ovim je postignuta veoma velika fleksibilnost pošto je sada moguće odabrati minimalan skup funkcionalnosti jezgra koje ispunjavaju specifikacije sistema. Ako u toku projektovanja dođe do izmene specifikacija sistema moguće je projektovani sistem brzo prilagoditi novim specifikacijama reprogramiranjem odgovarajućih delova sistema. Prednost ovih sistema je i ta što su u celosti opisani u HDL kodu. Ovo omogućava prebacivanje celokupnog dizajna na novije, brže FPGA čipove (koji nisu ni postojali u trenutku projektovanja sistema) čime se sam projekat štiti od zastarevanja. Kako je *soft-core* procesor raspoređen po resursima FPGA čipa, to njegove električne karakteristike variraju zavisno od implementacije, ima veće zauzeće prostora, veću potrošnju energije i manju brzinu od *hard-core* procesora iste konfiguracije. S*oft-core* procesori se veoma retko koriste u

M. Novaković, Elektrotehnički fakultet u Beogradu, Srbija (e-mail: novakovic@el.etf.rs).
D. El Mezeni, Elektrotehnički fakultet u Beogradu, Srbija (e-mail: elmezeni@el.etf.rs).
L. Saranovac, Elektrotehnički fakultet u Beogradu, Srbija (e-mail: laza@el.etf.rs).

punoj konfiguraciji tako da su u konkretnim aplikacijama češće dosta povoljnije rešenje od *hard-core* procesora. Vodeći proizvođači FPGA čipova nude svoje varijante *soft-core* procesora optimizovanih za arhitekturu svojih čipova. Tako *Xilinx* nudi PicoBlaze i MicroBlaze procesorska jezgra, *Altera* nudi Nios II *soft-core* procesor, *Atmel* je u saradnji sa ARMom razvio Cortex-M1, prvi *soft-core* procesor zasnovan na ARM arhitekturi. Nabrojana procesorska jezgra spadaju u domen komercijalnih jezgara, pošto je za njihovo korišćenje potrebno kupovati odgovarajuće licence. Na tržištu postoji i veliki broj besplatnih procesorskih jezgara (*open-source*) kao što su LatticeMico8, LatticeMico32, OpenSPARC T1 i LEON3. *OpenCores* zajednica [1] ima za cilj razvoj hardvera na sličnim principima na kojima je zasnovana *open-source* zajednica. Trenutno je težište na razvoju digitalnih modula (jezgara) koji se mogu implementirati na FPGA čipovima. Lista jezgara koji su nastali iz *OpenCores* inicijative je zaista impresivna, a najpoznatije je *OpenRISC* procesorsko jezgro.

III. EMBEDDED LINUX OPERATIVNI SISTEM

Glavne prednosti *embedded* Linux operativnog sistema proizilaze iz njegove *open-source* filozofije. Postoji mnoštvo besplatnih distribucija, alata i gotovih rešenja. Podržano je preko 25 tipova arhitekture i mnoge konkretne platforme. Neke od najpoznatijih Linux distribucija za *embedded* uređaje su *BlueCat*, *Denx ELDK*, *MontaVista*, *TimeSys* i za *soft-core* IP *Petalogix* baziran na *uClinux*-u. Odabirom komercijalnog operativnog sistema poslovni uspeh korisnika se neraskidivo vezuje za proizvođača operativnog sistema. *Embedded* Linux donosi malu zavisnost od proizvođača. Korisnici komercijalnih operativnih sistema su prinuđeni da čekaju proizvođače da razviju drajvere za sve nove uređaje. Otvorenost koda i veliki broj programera koji doprinose Linux operativnom sistemu znače podršku za najnoviji hardver.

Vreme razvoja proizvoda (*time to market*) je jedno od osnovnih faktora uspešnosti proizvoda. Izbor Linux-a kao operativnog sistema za proizvod dozvoljava da se većina aplikacija razvija na kućnim računarima (sa Linux distribucijom) a zatim relativno lako premesti na ciljni uređaj. Takođe postoji i veliki broj gotovih i besplatnih aplikacija.

Otvorenost Linux koda za javnost doprinela je kvalitetu, stabilnosti i pouzdanosti koda [2]. Linux programski kod je modularan i dobro struktuiran, čime odabir samo neophodnih funkcionalnosti za *embedded* sistem postaje lakši.

Velika zajednica Linux programera je verovatno najveća prednost *embedded* Linux-a. Vrlo često će na specijalizovanim Internet forumima stručnjaci odgovoriti na pitanje, odnosno dati pomoć u rešavanju problema.

Pored nabrojanih prednosti, najveća mana *embedded* Linux-a je sporiji rad i veće zauzeće resursa u nekim slučajevima, što je posledica kompleksne strukture Linux-a i visokog nivoa specijalizovanog znanja potrebnog da bi se Linux konfigurisao da radi sa minimumom potrebnih resursa.

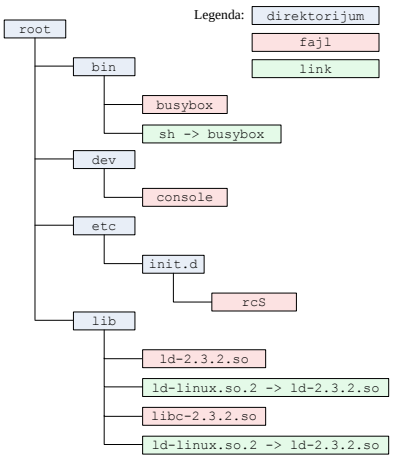
IV. IMPLEMENTACIJA LINUX OPERATIVNOG SISTEMA

Implementacija Linux operativnog sistema na *embedded* uređajima najčešće podrazumeva razvoj, testiranje i *debug*-ovanje programa na računaru opšte namene (*host*) koji je sa *embedded* uređajem (*target*) povezan nekom vrstom veze. Najčešći tipovi veza i njihove namene su serijska veza (konzolni ulaz/izlaz), JTAG (*debug*-ovanje, konfigurisanje i programiranje) i Ethernet (*debug*-ovanje, mrežni fajl sistem). *Target* uređaju moraju biti, u opštem slučaju, dostupni izvršni fajl operativnog sistema (*kernel image*), *bootloader*, fajl sistem i u slučaju FPGA čipova konfiguracioni fajl hardvera (BIT fajl). Razvoj softvera *embedded* Linux sistema podrazumeva konfigurisanje i prevođenje (*compile*-iranje) Linux jezgra *(kernel*-a), razvoj fajl sistema, podešavanje inicijalizacije sistema i razvoj aplikacija.

Konfigurisanje *kernel*-a je u opštem slučaju kompleksna radnja i podrazumeva podešavanje svih parametara vezanih za konkretan hardver, kao i pisanje ili prepravljanje inicijalizacionih delova koda. Najznačajniji korak konfiguracije je definisanje spiska hardverskih periferija sa njihovim adresama i eventualnih *driver*-a periferija. Konfigurisani *kernel* je potrebno prevesti u izvršni *kernel image*. Kako su po pravilu *host* i *target* uređaji različite arhitekture, za proces prevođenja potrebno je koristiti *cross-compiler*. To je prevodilac koji se izvršava na arhitekturi *host* računara, ali koji kao izlaz ima izvršne fajlove za arhitekturu *target* uređaja.

Kao *bootloader* se najčešće koriste gotova rešenja kako bi se smanjila mogućnost da se javi greška prilikom osnovne inicijalizacije hardvera sistema. Time se proces ispravljanja grešaka može fokusirati na greške prilikom inicijalizacije *kernel*-a. *Bootloader* treba da učita *kernel image* iz memorije *target* uređaja ili sa mrežne lokacije, eventulano ga otpakuje i preda kontrolu *kernel*-u. Vrlo popularan je univerzalni *bootloader Das U-Boot* [3].

Razvoj fajl sistema podrazumeva odabir potrebnih biblioteka, sistemskih i uslužnih programa, njihovo *cross*-kompajliranje i smeštanje na odgovarajući medijum. Većina distribucija sadrži već kompajlirane velike fajl sisteme iz kojih je potrebno odabrati samo željene delove. Fajl sistem može biti smešten na fizičku memoriju *target* sistema ili može biti učitan sa mrežne lokacije (*Network File System*). Posebna vrsta fajl sistema je *initrd image*, koji predstavlja fajl sistem koji se nalazi u RAM memoriji. Izbor tipa fajl sistema i sadržaja fajl sistema mogu biti ključni faktori u dizajnu *embedded* sistema. Od ovih odluka zavisi potrebna količina memorije, brzina inicijalizacije sistema, brzina samog sistema, pouzdanost i trajnost podataka, potrošnja energije i slično. Zgodno rešenje za kompaktan fajl sistem predstavlja aplikacija *BusyBox* [4]. Ona predstavlja jedan izvršni fajl i nekoliko konfiguracionih fajlova koji zamenjuju veliki broj standardnih sistemskih aplikacija. Pozivi ka standardnim sistemskim aplikacijama se preusmeravaju na *busybox* izvršni fajl. Izgled minimalnog fajl sistema zasnovanog na bazi *BusyBox*-a prikazan je na Sl. 1.



Sl. 1. Minimalan *BusyBox* fajl sistem

Prikazani fajl sistem je veličine manje od 2MB.

V. IMPLEMENTACIJA PERIFERIJA FPGA SISTEMA

Da bi se pojednostavilo kreiranje kompleksnih *embedded* sistema postoje razvojni alati koji automatizuju većinu koraka projektovanja. Svi ovi alati su specifične arhitekture, ali pružaju slične pogodnosti. Osnovne mogućnosti ovih alata biće prikazane na primeru *Xilinx*-ovih alata.

Integrated Software Environment (ISE) [5] predstavlja temelj svakog *Xilinx* FPGA dizajna. Kako je u procesu projektovanja potrebno programirati FPGA čip mnogi koraci kao što su rutiranje čipa, vremenska analiza i programiranje čipa su integrisani unutar ISEa.

EDK [6] predstavlja skup alata i IP-ova koji omogućavaju kompletan razvoj *embedded* procesorskog sistema za implementaciju na *Xilinx* FPGA čipovima. EDKu je potreban ISE za sintezu mikroprocesorkog dizajna, mapiranje dizajna na FPGA čip i za generisanje i spuštanje BIT fajla. Dve osnovne komponente EDK su *Xilinx Platform Studio* (XPS) koji služi za razvoj hardvera i *Software Development Kit* (SDK) koji služi za razvoj softvera.

Hardeverska platforma predstavlja fleksibilan *embedded* procesorski sistem koji se kreira za traženu namenu. Hardverska platforma se sastoji od jednog ili više procesora i periferija povezanih na procesorske magistrale. EDK čuva opis hardverske platforme u *Microprocessor Hardware Specification* (MHS) fajlu [7]. Za razvoj hardverske platforme koristi se XPS.

Softverska platforma predstavlja skup drajvera i opciono operativnog sistema u kojem se kreiraju aplikacije. Uključene su samo one bibilioteke koje se koriste u konkretnom dizajnu. EDK čuva opis softverske platforme u *Microprocessor Software Specification* (MSS) fajlu [7]. Jednostavnije aplikacije mogu biti projektovane i direktno u XPSu dok se za ozbiljnije programe i debagovanje koristi SDK.

Po završetku projektovanja hardverske i softverske platforme kreira se BIT fajl (mapa veza u FPGA čipu) za odgovarajući FPGA čip.

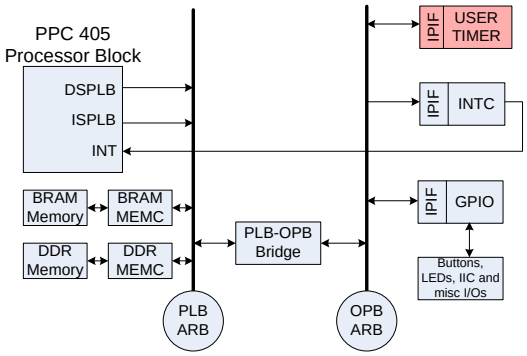
EDK pruža alate za verifikaciju hardvera i softvera. Za verifikaciju hardvera može se kreirati simulacioni model i pokrenuti na HDL simulatoru. XPS automatski podešava parametre potrebne za verifikaciju uključujući i HDL fajlove za simulaciju. Može se birati između modela ponašanja, strukturalnog i vremenski preciznog modela. Za verifikaciju softvera može se učitati dizajn na razvojnu ploču i odatle debagovati ili se može pokrenuti na virtuelnoj platformi (kompjuter na kom se razvija) korišćenjem simulatora instrukcijskog seta. Ovo omogućava paralelan razvoj hardvera i softvera *embedded* sistema čime se znatno smanjuje vreme potrebno za proces projektovanja.

Kako se u samom FPGA čipu ostvarivanjem odgovarajućih veza može implementirati funkcionalnost korisničkih periferija, nema potrebe za dodavanjem novog hardvera, već je dovoljno konfigurisati već postojeći tako da odgovara nameni.

Jedan od načina za upravljanje periferijom je pisanje *standalone* aplikacija korišćenjem SDKa i integrisanjem ove aplikacije u izvršni ELF fajl. Drugi pristup je korišćenje Linux operativnog sistema koji olakšava izvršavanje aplikacija.

VI. REALIZACIJA SISTEMA

Radi potvrde ispravnosti prethodno navedenih koraka i u želji da se konkretan sistem sa opisanim procedurama koristi u edukativne svrhe, autori su realizovali sopstveni sistem, prikazan na slici 2.



Sl. 2. Blok šema sistema

Konfiguracija hardvera sistema je zasnovana na referentnom dizajnu za *Xilinx* ML403 platformu [8]. Platforma sadrži Virtex-4 FX FPGA čip sa *hard-core* IP PowerPC procesorom. Pored elementa referentnog dizajna, implementirana je tajmerska, brojačka, periferija sa promenljivom radnom učestanošću.

Za povezivanje sa periferijama PowerPC koristi IBM *CoreConnect* arhitekturu koja u sebi uključuje PLB (*Processor Local Bus*) i OPB (*On-Chip Pheripheral Bus*) magistrale. Procesor i RAM memorije su povezani direktno na PLB magistralu koja je brza, dok su ostale periferije povezane na sporiju OPB magistralu. OPB magistrala ima jednostavnije protokole i bolje podržava veći broj uređaja od PLB magistrale. Za povezivanje na obe magistrale postoji jedinstveni IPIF (*intellectual property interface*) interfejs koji omogućava projektovanje periferije nezavisno od toga kako će ona biti povezana u

sistem i bez poznavanja protokola magistrale. IPIF interfejs takođe omogućava jednostavno premeštanje periferije sa jedne na drugu magistralu bez izmene same periferije. Konfigurisanje IPIF interfejsa je automatizovano u XPSu. Autori su korišćenjem alata konfigurisali svoju periferiju za povezivanje na OPB magistralu. Nakon generisanja IPIF interfejsa, modifikovan je fajl kojim se opisuje konkretna funkcionalnost periferije (*user_logic.vhd*). Takođe su automatski generisane adrese registara periferije pomoću XPSa. Opis hardverskog dela sistema, neophodan za kofigurisanje FPGA čipa, sadržan je u automatski generisanom BIT fajlu.

Neophodnu spregu između hardvera i softvera predstavlja BSP (*board support package*) koji XPS takođe automatski generiše. U sklopu BSPa se nalaze drajveri implementiranih perferija i informacije o adresama. Informacije o adresama neophodne za konfigurisanje programskog okruženja se nalaze u fajlu *xparameters.h*.

Kao *cross-compiler* autori su iskoristili gotov prevodilac za PowerPC 405 arhitekturu koji se nalazi u sklopu *Denx* ELDK distribucije.

Implementirana je verzija 2.6.25.4. Linux jezgra (*kernel*-a). *Kernel* je konfigurisan polazeći od standardne konfiguracije za ploču ML403. Modifikacijom standardne konfiguracije je specificiran način *boot*-ovanja *kernel*-a, konzolna komunikacija serijskim putem i naznačen konkretan *cross-compiler* koji se koristi. Uključivanjem generisanog *xparameters.h* fajla *kernel* dobija informacije o periferijama i adresama prisutnim u sistemu. Kako je korišćena verzija *kernel*-a sadržala generičke drajvere za razvojnu ploču, nije bilo potrebno koristiti drajvere generisane XPSom u okviru BSPa. Nakon završene konfiguracije *kernel*-a i njegovog prevođenja dobija se ELF fajl. U njemu su definisani kod, podaci, memorijska mapa sistema i u nekim slučajevima jednostavan *bootstrap loader*.

Kao način za podizanje sistema (*boot*-ovanja) autori su se odlučili za *Xilinx*-ov *SysAce*. On omogućava istovremeno konfigurisanje FPGA čipa i operativnog sistema tokom procesa *boot*-ovanja sa memorijske *flash* kartice. Potrebne informacije koje *SysAce* koristi da bi konfigurisao sistem se nalazi u ACE fajlu. XPS pruža mogućnost automatskog generisanja ACE fajla korišćenjem BIT i ELF fajlova.

Za fajl sistem iskorišćen je jedan od gotovih fajl sistema u okviru *Denx* ELDK distribucije zasnovan na *BusyBox*-u. Fajl sistem je smešten na istoj *flash* kartici kao i ACE fajl. Modifikovanjem fajl sistema je omogućeno korišćenje *GCC* prevodioca [10] na samom *target* uređaju.

Komunikacija između *target* i *host* uređaja obezbeđena je korišćenjem konzolne aplikacije *minicom* na strani *host*-a koja serijskim putem komunicira sa *target* uređajem. Ovo omogućava zadavanje komandi *target* uređaju preko *host* računara i prikaz informacija sa *target* uređaja direktno na *host* računaru.

Upravljanje implementiranom periferijom, promena učestanosti i praćenje stanja brojača, obezbeđeno je dvema nezavisnim korisničkim aplikacijama napisanim u *C* programskom jeziku. Uz posredovanje Linux operativnog sistema, obe aplikacije pristupaju memorijskim mapiranim registrima periferije. Aplikacije su pokrenute kao procesi u okviru Linux operativnog sistema, čime im je omogućeno konkurentno izvršavanje.

ZAKLJUČAK

Osnovni problem koji su autori rešavali je definisanje procedura i postupaka za brži razvoj *embedded* sistema na bazi postojećih operativnih sistema, prvenstveno Linux-a, i postojećih programabilnih čipova. Na žalost ove procedure za konkretne slučajeve nisu definisane tako da su autori u sklopu rada imali veliki broj bezuspešnih pokušaja implementacije sistema. Primetna je velika razlika između teorijskog opisa osnovnih koraka pri dizajnu i praktične implementacije ovih koraka. Zbog ovoga u svetu postoji mnogo specijalizovanih foruma gde se razmenjuju stečena iskustva [9] na konkretnim problemima.

Na osnovu opisanih i definisanih procedura autori su realizovali jedan namenski SoC sistem. Osnovna namena realizovanog računarskog sistema, zajedno sa opisanim procedurama i postupcima, je edukativnog karaktera. Rad je proistekao iz želje autora da sistematizuju stečena znanja i omoguće narednim generacijama studenata brže uključivanje u savremeni inženjerski pristup rešavanju problema.

ZAHVALNICA

Autori se zahvaljuju kolegama iz firme „Bitgear Wireless Design Services“ iz Beograda na stručnoj podršci.

LITERATURA

[1] *OpenCores*, www.opencores.org/
[2] *The Linux Kernel Archives*, kernel.org.
[3] *Das U-Boot - Universal Bootloader*, sourceforge.net/projects/u-boot
[4] *BusyBox*, www.busybox.net
[5] *Xilinx ISE 9.2i Software Manuals and Help*, Xilinx, 2007
[6] *Embedded System Tools Reference Manual*, Xilinx, 2007
[7] *Platform Specification Format Refernce Manual*, Xilinx, 2007
[8] *ML403 Reference design*, *www.xilinx.com/products/boards/ml403/reference_designs.htm*
[9] *Linux on Embedded PowerPC Developers*, https://ozlabs.org/mailman/listinfo/linuxppc-embedded
[10] *GCC, the GNU Compiler Collection*, gnu.gcc.org

ABSTRACT

Embedded FPGA systems with Linux operating system are very flexible and provide rapid development. Overview of common processor cores suitable for implementation in programmable components, as well as most important Linux characteristics is presented. This paper describes basic steps and tools used in the system design based on ML403 development board with Virtex-4 FX FPGA chip containing PowerPC processor core.

EMBEDDED FPGA LINUX SYSTEMS

Dragomir El Mezeni, Milan Novaković
Mentor: Lazar Saranovac, PhD