FPGA implementacija 16-bitnog RISC procesora sa fon-Nojmanovom arhitekturom

Stevan D. Marinković, Nebojša Z. Ćirović, Dragoljub S. Petrović

Mentori: prof. dr Jelena Popović, prof. dr Lazar Saranovac

***Sadržaj* — Realizacija 16-bitnog RISC procesora sa fon-Nojmanovom arhitekturom u VHDL-u i implementacija pomenutog procesora na Spartan3E čipu je osnovna tema ovog rada. RISC procesori imaju redukovan set instrukcija, što omogućava jednostavniju implementaciju. Opisana je arhitektura realizovanog procesora. Objašnjen je način realizacije u VHDL-u, sa posebnim osvrtom na organizaciju koda, realizaciju pojedinih celina, i korišćene alate za projektovanje i testiranje.**

***Ključne reči* — Fon-Nojmanova arhitektura, FPGA implementacija, mikroarhitektura VHDL koda, RISC procesor.**

I. UVOD

TERMIN procesor najčešće asocira na personalne računare. Međutim veliki broj današnjih uređaja za svakodnevnu upotrebu se ne može zamisliti bez ugrađenog jednog ili više procesora. U velikoj većini takvih primena korisniku je bitna namena tog uređaja a ne i procesor koji se nalazi unutar uređaja. Radi jednostavnosti, pouzdanosti i cene često je uređaj realizovan pomoću mikrokontrolera. Postoji bitna razlika između procesora za personalne računare i procesora koji se ugrađuju u takve namenske sisteme. Procesori za personalne računare imaju znatno veći broj instrukcija i komplikovaniji su za implementaciju u odnosu na procesore u namenskim sistemima. Osnovna ideja čitavog projekta bila je da se projektuje procesor koji je pogodan za implementaciju na FPGA čipu, RISC arhitekture radi jednostavnosti instrukcijskog seta. Takođe projektovani procesr mora imati mogućnost nadogradnje pojedinih periferija, u cilju dobijanja kompletnog mikrokontrolerskog sistema na FPGA čipu. VHDL implementacija je jedna od tri podjednako važne etape u projektovanju i testiranju RISC procesora. Detaljan opis projektovanja hardvera za ovaj procesor može se pronaći u diplomskom radu Nebojše Ćirovića [1], više detalja o FPGA implementaciji u diplomskom radu Stevana Marinkovića [2], dok se opis realizacije asemblerskog prevodioca nalazi u diplomskom radu Dragoljuba Petrovića [3]. Projektovani procesor ima identičan set instrukcija kao procesori kompanije *Texas Instruments* iz serije MSP430. *Data sheet* za pomenutu familiju procesora je poslužio kao smernica u izboru seta instrukcija, načina adresiranja i za uvid u registre koje poseduju procesori iz serije MSP430 [4]. Procesor je realizovan korišćenjem *Xilinx*-ovog paketa ISE 8.1i. Za testiranje je korišćen ModelSim XE III 6.1e simulator.

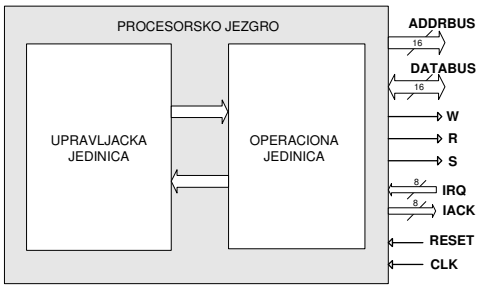
Stevan D. Marinković, Elektrotehnički fakultet u Beogradu, Srbija (telefon: +381-64-1852528; e-mail: stevankg2004@yahoo.com).

Nebojša Z. Ćirović, Elektrotehnički fakultet u Beogradu, Srbija (telefon: +381-64-2173130; e-mail: nebojsachiro@yahoo.com).

Dragoljub S. Petrović, Elektrotehnički fakultet u Beogradu, Srbija (telefon: +381-11-602100; e-mail: srbapet@gmail.com).

II. OPIS ARHITEKTURE PROCESORA

Projektovani procesor ima fon-Nojmanovu arhitekturu [1]. Poseduje zajednički memorijski prostor za instrukcije i podatke. Adresna magistrala je 16-bitna, što znači da je memorijski prostor veličine 64 KB. Magistrala podataka je takođe šesnaestobitna. Pored adresne magistrale i magistrale podataka, projektovani procesor poseduje i dve interne magistrale [1], u cilju olakšanja komunikacije između pojedinih registara. Procesor poseduje četiri banke po 12 opštenamenskih registara (ukupno 48 opštenamenskih registara), što je osnovna razlika u odnosu na procesore iz serije MSP430 [4]. Ova osobina omogućava smeštanje velikog broja podataka u registre, i brzo izvršavanje instrukcija. Registri specijalne namene su: programski brojač (PC), statusni registar (PSW), pokazivač steka (SP) i generator konstanti (CG). Podržano je 27 ugrađenih instrukcija i 7 načina adresiranja. Pored ugrađenih instrukcija, podržan je i rad sa emuliranim instrukcijama [1]. Emulirane instrukcije se dobijaju kombinovanjem odgovarajućih ugrađenih instrukcija. Moguć je pristup 8-bitnim i 16-bitnim periferijama i direktan transfer podataka iz memorije u memoriju, bez posredstva nekog od registara. Za obradu prekida, implementiran je mini kontroler prekida sa 8 ulaza [1]. Jedan ulaz je predviđen za nemaskirajuće prekide, dok su ostalih 7 poređani po prioritetu, i mogu se maskirati odgovarajućim bitima u statusnom registru. Blok šema procesorskog jezgra je prikazana na Sl. 1.



Sl. 1. Struktura procesorskog jezgra

Sa Sl. 1. se može videti da se procesor sastoji iz operacione i upravljačke jedinice, čija je realizacija opisana u četvrtom i petom poglavlju ovog rada. Grubo gledano, VHDL kod je takođe organizovan na način prikazan na Sl. 1, s tim što se operaciona i upravljačka jedinica sastoje iz nekoliko manjih celina, o čemu će biti više reči u poglavljima 4 i 5.

III. ARITMETIČKO-LOGIČKA JEDINICA

A. *Opis aritmetičko-logičke jedinice*

Aritmetičko-logička jedinica (ALU) je jedan od osnovnih delova svakog procesora. Osnovna namena aritmetičko-logičke jedinice je izvršavanje aritmetičkih i logičkih operacija procesora. Pregled operacija koje se izvršavaju pomoću atitmetičko-logičke jedinice projektovanog proceora dat je u tabeli 1. Za svaku operaciju dat je kratak opis.

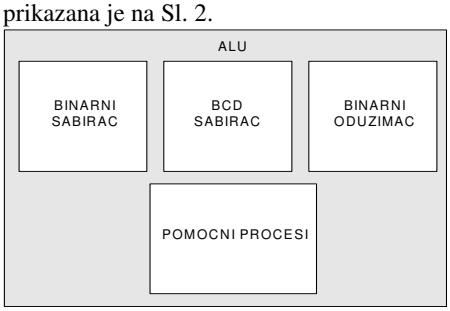
TABELA 1: ARITMETIČKO-LOGIČKE OPERACIJE PROCESORA.

Operacija	**Opis**
ADD(.B) src,dst	src + dst → dst
ADDC(.B) src,dst	src + dst + C → dst
DADD(.B) src,dst	src+dst+C → dst(decimalno)
SUB(.B) src,dst	dst + .not.src + 1 → dst
SUBC(.B) src,dst	dst + .not.src + C → dst
BIC(.B) src,dst	.not.src .and. dst → dst
BIS(.B) src,dst	src .or. dst → dst
AND(.B) src,dst	src .and. dst → dst
XOR(.B) src,dst	src .xor. dst → dst
RRA(.B) dst	Aritmetičko rotiranje u desno
RRC(.B) dst	Rotiranje u desno (na mesto najvišeg bita se stavlja bit C)
SWPB dst	Zamena mesta bajtovima u 16-bitnoj reči
SXT dst	Ekstenzija znaka

Oznaka *(.B)* u tabeli 1 označava da se operacija može izvršavati kako nad operandima dužine reči (16 bita), tako i nad bajtovima. Oznake *src* i *dst* označavaju izvorišni i odredišni operand, respektivno, dok je *C* oznaka za bit prenosa (*carry bit*).

B. *Opis VHDL modela aritmetičko-logičke jedinice*

Aritmetičko-logička jedinica je realizovana u VHDL-u instanciranjem i povezivanjem tri već isprojektovane komponente (binarni sabirač, binarni oduzimač i BCD sabirač) u višem nivou hijerarhije [5]. Ostale operacije su realizovane pisanjem odgovarajućih procesa (na sl. 2. označenih kao "Pomoćni procesi"), pošto su te operacije jednostavne i lako se implementiraju. Blok šema organizacije VHDL koda pisanog za aritmetičko-logičku jedinicu prikazana je na Sl. 2.



Sl. 2. Mikroarhitektura VHDL koda za aritmetičko-logičku jedinicu

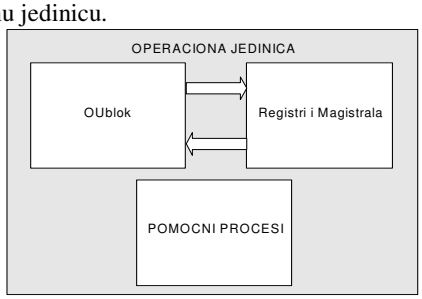
IV. OPERACIONA JEDINICA

A. *Opis operacione jedinice*

Operaciona jedinica sadrži hardver koji podržava izvršavanje instrukcija projektovanog procesora. Osnovne komponente operacione jedinice su aritmetičko-logička jedinica, programski brojač, stek pointer, statusni registar, instrukcijki registar, registar za prihvatanje adresa (MAR), registar za prihvatanje podataka (MBR) i četiri banke opštenamenskih registara. Izbor banke registara se vrši u statusnom registru selekcijom odgovarajućih bita. Pošto je u projektu procesora predviđen *pipeline* dubine 1 [1] (metod za ubrzanje izvršavanja instrukcija), dodat je pomoćni instrukcijski registar (PIR). U PIR se unapred dovode instrukcije, kako bi one bile spremne za izvršavanje odmah nakon završetka instrukcije koja im prethodi, bez prethodnog „dohvatanja" instrukcije iz memorije. Signale koji kontrolišu hardver operacione jedinice generiše upravljačka jedinica. O tome će biti više reči u narednom poglavlju.

B. *Opis VHDL modela operacione jedinice*

Operaciona jedinica je u VHDL-u realizovana na taj način što su najpre realizovane sve manje komponente koje ulaze u sastav operacione jedinice (registri, multiplekseri, ALU,...). Zatim su te komponente instancirane i povezane u višem nivou hijerarhije. Na Sl. 3. prikazan je blok dijagram mikroarhitekture VHDL koda za operacionu jedinicu.



Sl. 3. Mikroarhitektura VHDL koda za operacionu jedinicu

Formalno gledano, operaciona jedinica se sastoji iz dva veća bloka. To su blokovi „Oublok" i „Registri i

magistrala“. Podela na ova dva bloka nije urađena prema njihovoj funkciji, već prema obimu hardvera koji se u njima nalazi. U bloku „Registri i magistrala“ implementirani su opštenamenski registri (4 banke po 12 registara) i dve interne magistrale na koje izlaze ti registri. „Oublok“ sadrži implementaciju ostalih registara (MBR, MAR, IR, PR, PIR, PC, SP, PSW) [1]. U „Oublok“-u je instancirana i aritmetičko-logička jedinica i implementirane adresna magistrala i magistrala podataka.

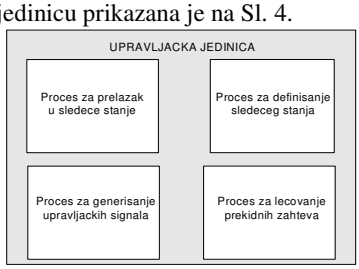
V. UPRAVLJAČKA JEDINICA

A. *Opis upravljačke jedinice*

Upravljačka jedinica je blok pocesora u kome se vrši generisanje neophodnih signala koji kontrolišu hardver operacione jedinice. Pri projektovanju i implementaciji upravljačke jedinice mora se biti posebno obazriv, jer i najmanja greška u generisanju signala može dovesti do pogrešnog i nepredvidljivog rada procesora. Upravljačka jedinica projektovanog RISC procesora je realizovana kao sinhrona mašina stanja [1], koja se sastoji od 37 stanja. U svakom stanju se u zavisnosti od tekuće instrukcije i načina adresiranja vrši generisanje odgovarajućih upravljačkih signala i definisanje sledećeg stanja u koje treba preći.

B. *Opis VHDL modela upravljačke jedinice*

S obzirom na veliki broj signala koje je potrebno generisati i na kompleksnost upravljačke jedinice, realizacija mašine stanja u VHDL-u je rađena iz nekoliko koraka. Najpre je napravljana i testirana „prazna“ mašina stanja. U „praznoj“ mašini stanja su definisani samo prelazi iz stanja u stanje. Zatim su u svim stanjima osim u stanju u kome se tumači instrukcijska reč definisani upravljački signali. Tako dobijena mašina stanja je testirana, kao bi se uverili u njenu ispravnost. Najzad su upravljački signali definisani i u stanju u kome se tumači instrukcijska reč. Ovo stanje je najkompleksnije, jer za svaku instrukciju mora da pokrije sve načine adresiranja. Zbog ovoga je potrebno proći kroz veliki broj *if* i *case* uslova. To prouzrokuje generisanje kompleksne kombinacione logike. Mikroarhitektura VHDL koda za upravljačku jedinicu prikazana je na Sl. 4.



Sl. 4. Mikroarhitektura VHDL koda za upravljačku jedinicu

VHDL kod kojim je opisana mašina stanja sastoji se iz četiri procesa. Prvi proces je osetljiv na signal takta (CLK) i omogućava prelazak iz trenutnog u sledeće stanje. U drugom procesu se definišu prelazi iz stanja u stanje (za svako od 37 stanja). U trećem procesu su generisani upravljački signali za svako stanje. Ovaj proces je bilo najteže napisati zbog velikog broja signala, pa su u toku pisanja često vršene simulacije radi provere ispravnosti. Četvrti proces je najjednostavniji i služi za lečovanje prekidnog zahteva u slučaju da se mašina stanja nalazi u trideset četvrtom stanju [1].

VI. SIMULACIJA RADA PROJEKTOVANOG PROCESORA

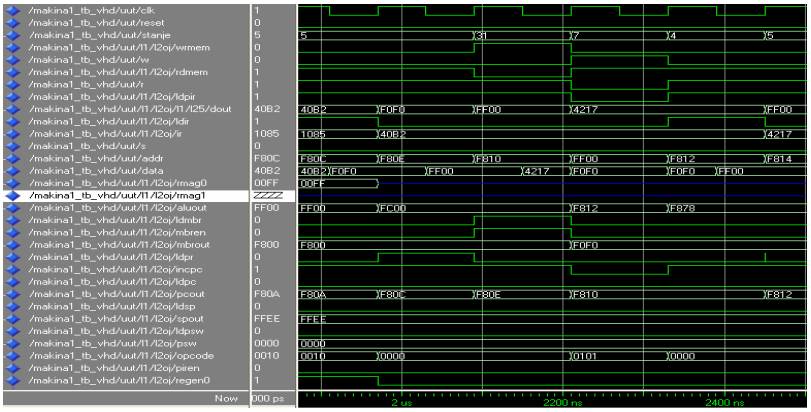
Nakog završenog projektovanja, bilo je potrebno verifikovati ispravnost rada procesora. Za potrebe simulacije rada projektovanog procesora instancirana je blok RAM memorija. Blok RAM memorija već postoji kao gotov blok na Spartan3E čipu. Kako bi se što jednostavnije, a što detaljnije testirao procesor, u asembleru je pisan kod iz koga je generisan bit fajl [3] sa sadržajem memorijskih lokacija u koje se smešta asemblerski program. Kodovi instrukcija, dobijeni generisanjem bit fajla su smeštani u blok RAM memoriju. To omogućava testiranje većeg broja instrukcija jednim pokretanjem simulatora. Testiranje je izvršeno korišćenjem ModelSim XE III 6.1e simulatora. Rađene su *post place and route* simulacije [5], koje su najpribližnije realnom ponašanju komponente. Radi ilustracije rezultata koji se dobijaju korišćenjem ModelSim XE III 6.1e simulatora, na Sl. 5. prikazani su vremenski dijagrami za instrukciju MOV #0xF0F0, &0xFF00. Ova instrukcija obavlja prenos neposrednog podatka na apsolutnu adresu u memoriji [1]. Instrukcija traje 4 takta i do izražaja dolazi *pipeline* [1]. Najpre se u pomoćnom instrukcijskom registru (PIR) nalazi neposredni podatak koji se šalje, a zatim i adresa na koju se podatak smešta. Ta adresa se zatim iz PIR-a šalje u MAR registar. Na magistrali podataka se u momentu upisa adrese u MAR nalazi podatak koji treba upisati (0xF0F0). Generisanjem signala *wrMEM* (tj. signala *W* koji se dobija lečovanjem signala *wrMEM* jedan takt kasnije) vrši se upis podatka sa magistrale podataka na unapred generisanu apsolutnu adresu u memoriji.

VII. TESTIRANJE NA RAZVOJNOJ PLOČI

Poslednja etapa u testiranju projektovanog RISC procesora bila je testiranje na razvojnoj ploči. Procesor je implementiran u Spartan3E čipu na *Xilinx* razvojnoj ploči HW-SPAR3E-SK [6]. Korišćenjem periferija koje postoje na toj razvojnoj ploči, testirana je ispravnost i brzina rada procesora.

A. *Opis Spartan3E čipa*

Pre opisa testova koji su rađeni na samoj ploči, biće ukratko dat opis arhitekture Spartan3E čipa na kome je procesor implementiran. Spartan3E čip pripada familiji FPGA čipova koje proizvodi kompanija *Xilinx* i sastoji se od pet programabilnih funkcionalnih celina [7]. Konfigurabilni logički blokovi sadrže fleksibilne *Look-Up* tabele (LUT) za implementaciju logike, kao i memorijske



Sl. 5. Rezultati simulacije instrukcije MOV #0xF0F0,&0xFF00

elemente kao što su flip-flopovi i lečevi. Ulazno-izlazni blokovi (IOB) kontrolišu razmenu podataka između ulazno-izlaznih pinova i interne logike u čipu. Blok RAM omogućava skladištenje podataka u formi 18 Kbit-nih dual port blokova. Blokovi množača hardverski određuju proizvod dva osamnaestobitna binarna broja. Blokovi za distribuciju takta (*Digital Clock Manager* - DCM) omogućavaju kalibraciju, u potpunosti digitalni način distribucije, multipleksiranje, deljenje i fazno pomeranje signala takta.

B. Opis testova na razvojnoj ploči

Prvih nekoliko testova je urađeno sa prekidačima i LED diodama, koji postoje na razvojnoj ploči. Prekidačima i diodama su dodeljene adrese preko kojih se vrši pristupanje tim periferijama. Zatim je pisan asemblerski kod pomoću koga su očitavana stanja prekidača i uključivane LED diode u zavisnosti od stanja prekidača. Takođe su rađeni testovi u kojima je pisan asemblerski kod koji vrši određene operacije, a radi provere ispravnosti izvršenih instrukcija pojedini rezultati su upisivani u registar koji kontroliše rad LED dioda. Najopsežniji test procesora je urađen tako što je u asembleru pisan drajver za inteligentni LCD displej, a zatim vršen ispis na displej. S obzirom da inicijalizacija displeja zahteva veliki broj asemblerskih instrukcija, uspešnost ovog testa, verifikovana na Sl. 6. u velikoj meri potvrđuje da projektovani procesor radi onako kako je predviđeno. Treba još napomenuti da je testiranjem na ploči utvrđeno da procesor pouzdano radi na frekvenciji od 12.5 MHz.

Sl. 6. Ispis na LCD displej

VIII. ZAKLJUČAK

Jedna od najbitnijih informacija, kada je implementacija u pitanju, jeste informacija o zauzetosti resursa čipa. Iako čip na kome je implementiran procesor ne spada u red čipova sa velikim logičkim resursima, zauzetost logičkih resursa čipa je 36%, a zauzetost ulazno-izlaznih blokova samo 5%. To pruža mogućnost za dalji razvoj ovog procesora, ili za implementaciju odgovarajućih periferija, u cilju dobijanja kompletnog mikrokontrolerskog sistema na čipu. To je posebno korisno, zbog fleksibilnosti koju pruža jedan takav sistem.

LITERATURA

[1] N. Ćirović, "Projekat 16-bitnog RISC procesora sa fon-Nojmanovom arhitekturom" , diplomski rad
[2] S. Marinković "FPGA implementacija 16-bitnog RISC procesora sa fon-Nojmanovom arhitekturom" , diplomski rad
[3] D. Petrović, "Asemblerski prevodilac za 16-bitni RISC procesor sa fon-Nojmanovom arhitekturom" , diplomski rad
[4] Texas Instruments "*MSP430x44xx Family User's Guide"*, Vol 3. pp. 41-115
[5] Michael John Sebastian Smith, "*Application – Specific Integrated Circuits"*, Vol 10. pp. 380-478, Addison Wesely 1997.
[6] http://www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=HW-SPAR3E-SK-US-G
[7] Xilinx " *Spartan-3E FPGA Family Complete Data Sheet"*, available on http://direct.xilinx.com/bvdocs/publications/ds312.pdf

ABSTRACT

VHDL realization of 16-bits RISC processor with von Neumann architecture and implementation of the processor on Spartan3E chip is presented in this paper. RISC processors has reduced set of instruction, and that fact provide easier implementation. In this paper is explaied architecture of the implemented processor. We explained VHDL realization with special attention on code structure, realization of main units and software which we used for projecting and testing the processor.

FPGA IMPLEMENTATION OF 16-BITS RISC PROCESSOR WITH VON NEUMANN ARCHITECTURE

Stevan Marinković, Nebojša Ćirović, Dragoljub Petrović