



**ELEKTROTEHNIČKI FAKULTET
BEOGRAD**

**VHDL implementacija 32-bitnog procesora opšte
namene**

studenti: Andrej Sulem
390/2010
Nenad Prahovljanovic
400/2010

profesor: dr Veljko Milutinović
asistent: Živojin Šuštran

broj osvojenih poena:

Beograd
19.6.2016

VHDL implementacija 32-bitnog procesora opšte namene

Andrej Sulem

email: sa100390d@student.efn.rs

Nenad Prahovljanovic

email: pn100400d@student.efn.rs

1. DEFINISANJE PROJEKTA

1.1 Uvod

Veliki broj danasnjih uređaja za svakodnevnu upotrebu se ne može zamisliti bez ugrađenog jednog ili više procesora.

Procesori su sve brzi, manji i komplikovaniji ali njihova osnova je ista. Da bismo shvatili osnovne principe rada, proučavamo jednostavnije modele.

1.2 Ciljevi projekta

Cilj projekta je implementacija 32-bitnog procesora opšte namene sa jednim jezgrom koji je povezan sa dve keš memorije, jednom za podatke, drugom za instrukcije. Ovim projektom studenti treba da steknu osnovno iskustvo u projektovanju procesora.

2. OPIS DIZAJNA

2.1 Zabeleške uz dizajn

Ovde navesti sve pretpostavke koje su uzete u obzir u projektovanju i opšte zabeleške.

Interfejs procesora prema okolini sadrži linije za komunikaciju sa keš memorijama, RESET signal i signal kloka. Prilikom startovanja sistema RESET signal se postavlja na aktivnu vrednost i nakon nekog vremena se vraća na neaktivnu vrednost. Trajanje RESET signala je dovoljno da se ceo sistem resetuje. Jezgro je RISC arhitektura sa sledećim osobinama:

- registarski fajl
- load/store arhitektura
- adresiranje je bazirano na vrednostima u registrima
- uniformna instrukcijska rec, velicine 32bita

Procesor sadrži protočnu obradu. Broj stepeni je 5. Za instrukcije skoka se koristi prediktor skoka koji je realizovan po principu kesa sa 2-bitnom semom. Jezgro ima 32 registara opšte namene širine 32-bitna. Registri su obeleženi sa R0 – R31. Registri specijalne namene su: 1) PC je pokazivač na sledeću instrukciju i 2) SP pokazivač na vrh steka. SP pokazuje na prvu slobodnu lokaciju i stek raste ka nižim adresama.

2.2 Faze dizajna

Detaljno opisati sve faze dizajna i priložiti odgovarajuće dijagrame.

Procesor sadrži protocnu obradu. Postoje pet faza: FETCH, DECODE, EXE, MEM, WB.

2.2.1 Interfejs između procesora i memorije

Linije za komunikaciju sa keš memorijom podataka su:

- 1) 32 adresne linije,
- 2) 32 linije za čitanje podataka,
- 3) 32 linije za upis podataka i
- 4) kontrolne linije.

Linije za komunikaciju sa instrukcijskom keš memorijom su: 1) 32 adresne linije,

- 2) 32 linije za čitanje podataka i
- 3) kontrolne linije.

Adresibilna jedinica je reč. Reč je veličine 4 bajta.

2.2.2 Faze izvršavanja instrukcije

Instrukcije prolaze kroz pet faza procesora.

Faze su:

- 1.FETCH – dohvaćanje instrukcija
- 2.DECODE – dekodovanje instrukcija i citanje operanada iz registarskog fajla.
- 3.EXE – izvršavanje date instrukcije.
- 4.MEM – citanje i upis iz/u keš memoriju.
- 5.WB – upis u registarski fajl.

2.2.2.1 FETCH faza

U Fetch fazi se vrši dohvaćanje instrukcije. Prosledjuje se adresa instrukcijskom kesu i odatle se cita instrukcija. Adresa se salje i u prediktor skoka kako bi se izvršila provera da li je to instrukcija skoka. Ukoliko jeste, prediktor nam vraća predviđeni PC koji saljemo u instrukcijski kes i citamo instrukciju. Nakon toga, PC i instrukciju saljemo u narednu fazu.

2.2.2.2 DECODE faza

U ovoj fazi vršimo dekodovanje instrukcija, kao i citanje operanada iz registarskog fajla. Ka ovoj fazi se vrši i prosledjivanje iz narednih faza ukoliko se cita nekorektna vrednost iz registra. Kada se procita, vrednost registra, PC, neposredna vrednost i opcode se salju ka sledecoj fazi.

2.2.2.3 EXE faza

U EXE fazi na osnovu opcode-a izvršavamo datu instrukciju. U ovoj fazi je realizovana ALU jedinica na cijem izlazu dobijamo rezultat izvršavanja u zavisnosti od instrukcije. Ukoliko je instrukcija skoka, salju se potrebne informacije ka prediktoru radi azuriranja njegovog sadržaja. Izlaz iz ALU se vodi u sledeću fazu kao i adrese destinacionog i izvorisnog registra.

2.2.2.4 MEM faza

Mem faza cita i upisuje u data kes. Ukoliko se radi o instrukciji LOAD, cita se sa adrese koja je dobijena sa izlaza ALU jedinice i smesta u dest.registar koji se prosledjuje u narednu fazu. Ukoliko je store, upisuje se u memoriju iz izvorisnog registra na adresu koja se prosledjuje sa izlaza ALU jedinice.

2.2.2.5 WB faza

Write back faza vrši upis u registarski fajl. Ukoliko se radi o aritmeticko-logickoj instrukciji upisuje se rezultat operacije sa ALU jedinice. U slučaju da je load, upisuje se očitana vrednost iz data kesa.

3. IMPLEMENTACIJA

Jezik koji koristimo je VHDL (Very-high-speed-integrated-circuit Hardware Description Language) a alati Quartus II i ModelSim-Altera.

VHDL implementacija je jedna od tri podjednako vazne etape u projektovanju i testiranju RISC procesora.

Implementacija se sastoji iz dva glavna tipa logickih elemenata: kombinacionih i sekvencijalnih elemenata. Kombinacioni elementi su elementi koji rade sa podacima, znaci da njihovi izlazi zavise od tekućih ulaza. Sekvencijalni elementi su elementi koji drže neku vrednost. Svaki takav element ima najmanje dva ulaza i jedan izlaz. Dva ulaza su podatak koji treba da je upisan i clock signal. Izlaz obezbeđuje vrednost upisanu u prethodnom ciklusu.

3.1 Hazardi

Hazardi koji postoje su hazardi podataka.

3.2 Oporavak od hazarda

Hazard tipa RAW se može hardverski eliminisati tehnikom prosledjivanja. Ali postoje neke situacije RAW hazarda podataka kada je jedini način da se obezbedi korektno izvršavanje instrukcija u pipeline-u zaustavljanje pipeline-a. To se radi signalom 'stall'.

3.3 Primeri hazarda

Kao ilustracija hazarda podataka koji može da nastane kod pipeline izvršavanja instrukcija može se uzeti sledeći primer:

```
add R1, R2, R3
sub R4, R5, R1
and R6, R1, R7
or R8, R1, R9
xor R10, R1, R11
```

Sve instrukcije posle instrukcije add koriste rezultat instrukcije add koji se nalazi u registru R1.

Ovaj hazard se rešava prosledjivanjem. Add instrukcija već u EXE fazi ima vrednost registra R1 tako da se prosledjuje sub instrukciji u DECODE fazi prilikom citanja operanada. Takodje se iz MEM i WB prosledjuje u DECODE za instrukcije and i or respektivno.

Primer kada treba zaustaviti pipeline:

```
lw R1, 32(R6)
add R4, R1, R7
sub R5, R1, R8
and R6, R1, R7
```

Instrukcija lw ima podatak tek na kraju MEM faze, dok je add instrukciji taj podatak potreban u DECODE fazi. Jedini način je zaustavljanje pipeline. Sve instrukcije počev od instrukcije add su zakasnjene za jednu periodu signala takta. Sada je moguće proslediti podatak iz MEM faze u DECODE fazu add instrukcije. Takodje, sada instrukcija lw u prvoj polovini stepena WB upisuje u R1, a instrukcija sub cita u drugoj polovini stepena DECODE.

3.4 Prediktor skoka

Jedinica sa kesom predviđanja ima strukturu koja je jako slična strukturi kes memorije i realizovana je u tehnici asocijativnog preslikavanja. U asocijativnom delu se čuvaju adrese izvršavanih instrukcija skokova. U odgovarajućem ulazu RAM dela se nalazi predviđena vrednost za PC i bit predviđanja. Kad god se u stepenu IF

cita neka instrukcija adresa te instrukcije se vodi na ulaze asocijativnog dela kes memorije i vrsi provera da li se u nekom od ulaza asocijativnog dela nalazi ta vrednost. Ako se ne nalazi to znaci ili da nije instrukcija skoka ili je instrukcija skoka koja do sada nije izvsavana. Ako se nalazi, to znaci da je to instrukcija skoka koja je vec bila izvsavana i za nju postoji predvidjanje da li ce biti skoka ili ne. Predvidjanje se radi po dvobitnoj semi.

4. TESTIRANJE I VERIFIKACIJA

Celokupan sistem je testiran proveravanjem pojedinačnih instrukcija.

3.1 Test primer

Instrukcijski kes

00000001	
00000000	01010101010100000000000000000000
00000001	001100000010000000000000000010000
00000002	0011000001000000000000000000100000
00000003	100001000000000000000000100000000
00000004	10100111111000000000111111111110
00000005	11111100000000000000000000000000
00000100	00000100000000100000100010000000
00000101	00110000010000100000000000000001
00000102	00110100001000010000000000000001
00000103	10001000000000000000000000000000

Data kes init

00000000	00000000000000000000000000000000
00000001	00000000000000000000000000000000

Data kes ocekivano

000000A0	000000000000000000000000000010000
000000A1	00000000000000000000000000001111
000000A2	00000000000000000000000000001110
000000A3	00000000000000000000000000001101
000000A4	00000000000000000000000000001100
000000A5	00000000000000000000000000001011
000000A6	00000000000000000000000000001010
000000A7	00000000000000000000000000001001
000000A8	00000000000000000000000000001000
000000A9	00000000000000000000000000001111
000000AA	00000000000000000000000000001110
000000AB	00000000000000000000000000001011
000000AC	00000000000000000000000000001000
000000AD	00000000000000000000000000001111
000000AE	00000000000000000000000000001010
000000AF	00000000000000000000000000000001
000000B0	00000000000000000000000000000000

5. ZAKLJUČAK

Kada se napravljeni procesor istestira, sledeci korak je stavljanje na FPGA chip.

Pipelinig je standardna tehnika RISC procesora koja se koristi da unapredi kako clock speed tako i ukupne performanse. Pipelinig omogućava procesoru da radi na razlicitim stepenima obavljanja instrukcije u isto vreme tako da vise instrukcija moze da se izvrsi u kracem

vremenskom periodu. Kada je procesor sa pipeline-om u toku jednog clock-a svaki od tih modula, ili faza je u upotrebi u isto vreme izvsavajući različite instrukcije u paraleli.

6. LITERATURA

1. <http://home.etf.rs/~vm/os/vlsi/index.html>

2. http://rti.etf.bg.ac.rs/rti/ri3aor/literatura/predavanja/Pipeline_2007.pdf