POKAZNA VEŽBA 6

Strukture za računanje

Potrebno predznanje

- Urađena pokazna vežba 5
- Poznavanje aritmetičkih digitalnih sistema i aritmetičko-logičkih jedinica
- Osnovno znanje upravljačkih jedinica digitalnih sistema

Šta će biti naučeno tokom izrade vežbe?

Nakon urađene vežbe bićete u mogućnosti da:

- Projektujete strukture za računanje kao digitalne sisteme
- Projektujete upravljačke jedinice za strukture za računanje
- Razumete različite tipove upravljačkih jedinica i izaberete odgovarajući tip za datu primenu
- Projektujete upravljačke jedinice koje izvršavaju određeni program na datoj strukturi za računanje.

Apstrakt i motivacija

Digitalni sistemi su pokazali svoju pravu moć i pregršt mogućnosti primene onog trenutka kada su počeli da se primenjuju za računanje. Mašina za računanje složenih matematičkih operacija ubrzo je evoluirala u mašinu koja je postala svačiji lični asistent – obaveštava nas, omogućuje nam da pričamo sa udaljenim osobama, priča sa nama, planira nam dan, zabavlja nas, upravlja našim kućnim budžetom i asistira nas na još veliki broj načina. Naravno, još uvek može da rešava parcijalne diferencijalne jednačine, ukoliko vam ikad to zatreba. Ova mašina je, svakako, vaš omiljeni lični računar, a mozak te mašine je procesor – univerzalna struktura za računanje. U ovoj vežbi ćemo se upoznati sa strukturama za računanje, naučićemo kako se one projektuju i kako se njima upravlja. Kroz jednostavan primer, implementiraćete vašu prvu strukturu za računanje. Nakon ove vežbe bićete pripremljeni da rešite konačni zadatak našeg predmeta – projektovanje univerzalne strukture za računanje, procesora.

TEORIJSKE OSNOVE

1. Aritmetički elementi struktura za računanje

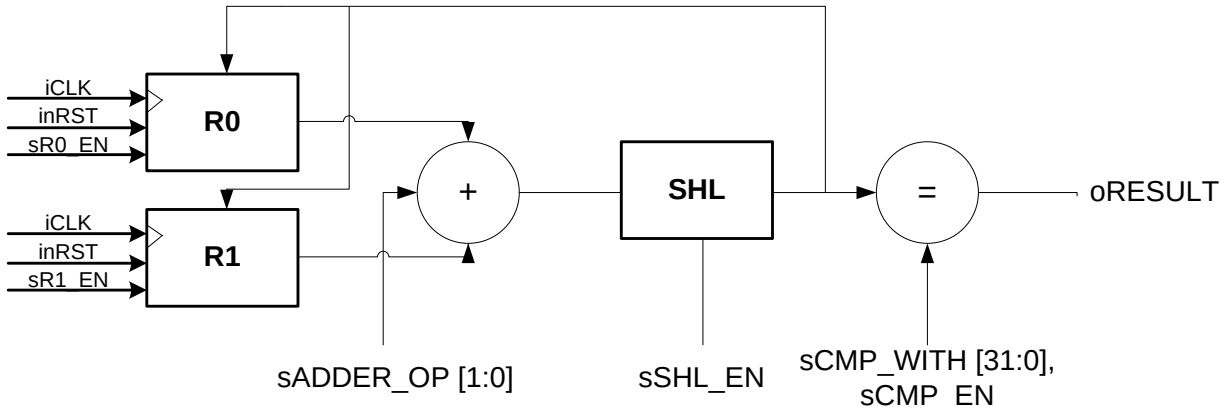
Strukture za računanje su digitalni sistemi projektovani s ciljem izvršavanja matematičkih aritmetičkih i logičkih operacija. Najkorišćenije komponente unutar struktura za računanje su standardne kombinacione i sekvencijalne mreže sa kojima ste već upoznati:

- Sabirači i oduzimači
- Množači i delioci
- Pomerači
- Komparatori
- Aritmetičko-logičke jedinice
- Multiplekseri
- Registri

Svaka struktura za računanje ima bar tri dela:

1. Registri opšte namene – za čuvanje izračunatih vrednosti
2. Aritmetičko-logički deo – za izvršavanje računanja
3. Upravljački deo – za upravljanje registrima i aritmetičko-logičkim delom strukture.

Primer dela strukture za računanje je dat na slici 1-1.



Slika 1-1. Primer aritmetičkog dela strukture za računanje

Struktura za računanje sa slike 1-1 se može analizirati kroz ranije definisane delove:

1. Registri opšte namene – R0 i R1 su dva 32-bitna registra, koriste se za čuvanje izračunatih vrednosti
2. Aritmetičko-logički deo se sastoji iz:
    a. 32-bitnog sabirača, koji prima dva 32-bitna operanda na ulazu i računa 32-bitni zbir prikazujući ga na izlazu (prenos se zanemaruje)
    b. 32-bitnog pomerača za jedno mesto u levo
    c. 32-bitnog komparatora, koji poredi dve vrednosti sa ulaza i na izlazu postavlja vrednost 1 ukoliko su ulazu jednaki, a 0 u suprotnom.
3. Upravljačka jedinica (nije prikazana na slici) kontroliše rad ostatka sistema:

a. Registri se kontrolišu pomoću dva signala dozvole upisa (jedan po registru); ukoliko je signal dozvole aktivan (na vrednosti 1), u registar se upisuje vrednost sa ulaza, u suprotnom registar pamti staru vrednost
b. Sabirač je kontrolisan 2-bitnim signalom sADDER_OP[1:0]
    i. Ukoliko je taj signal na vrednosti "00", prvi operand sabirača je vrednost registra R0, a drugi operand je vrednost registra R1
    ii. Ukoliko je taj signal na vrednosti "01", prvi operand sabirača je konstanta 1, a drugi operand je vrednost registra R1
    iii. Ukoliko je taj signal na vrednosti "10", prvi operand sabirača je vrednost registra R0, a drugi operand je konstanta 0
    iv. Ukoliko je taj signal na vrednosti "11", prvi operand sabirača je konstanta 0, a drugi operand je vrednost registra R1
c. Pomerač je kontrolisan pomoću signala dozvole pomeranja sSHL_EN; ukoliko je signal dozvole aktivan (na vrednosti 1), pomerač pomera ulaznu vrednost za jedno mesto ulevo, a u suprotnom ne pomera ulaznu vrednost
d. Komparator se kontroliše 32-bitnim operandom sCMP_WITH[31:0], u pitanju je vrednost sa kojom se poredi izlaz iz pomerača, kao i signalom dozvole poređenja sCMP_EN; ukoliko je signal dozvole neaktivan, komparator će na izlazu uvek davati vrednost 0 i neće vršiti poređenje.

Struktura za računanje neće vršiti nikakve operacije dok ne dobije aktivne kontrolne signale. Upravo pomoću ovih kontrolnih signala upravljačka jedinica može da kontroliše rad strukture za računanje – jednostavnim definisanjem vrednosti kontrolnih signala u svakom trenutku vremena. Okrećemo se sada projektovanju upravljačkih jedinica.

2. Upravljačke jedinice struktura za računanje

Upravljačka jedinica je centralna upravljačka stanica strukture za računanje. Njen zadatak je da definiše vrednosti kontrolnih signala čitavoj strukturi za računanje, za svaku komponentu. Upravljačke jedinice se mogu implementirati na dva načina:

- **Opšte upravljačke jedinice** se projektuju tako da mogu da izvršavaju proizvoljan skup operacija, tzv. program. Ove upravljačke jedinice primaju na ulazu *instrukcije*, tj. šifre koje govore upravljačkoj jedinici koju operaciju treba da izvrši u datom koraku. Instrukcije su često upamćene u memoriji iz koje se preuzimaju jedna za drugom i izvršavaju.
- **Specifične upravljačke jedinice** se projektuju tako da izvršavaju jedan predodređeni niz operacija. Ove upravljačke jedinice ne primaju instrukcije, nego prilikom svakog pokretanja generišu isti niz kontrolnih signala i time izvršavaju konstantan program. One su jednostavnije za projektovanje, ali ograničavaju mogućnosti sistema kojim upravljaju.

Prema tipu digitalnog sistema kojim je implementirana upravljačka jedinica, možemo razlikovati:

- **Kombinacione upravljačke jedinice** koje generišu kontrolne signale kombinacionom logikom, u zavisnosti od instrukcije koja se trenutno nalazi na njihovom ulazu. Ovim načinom se mogu implementirati jedino opšte upravljačke jedinice i one tada instrukcije izvršavaju uvek u jednom ciklusu takta sistema. *Specifične upravljačke jedinice se ne mogu implementirati na ovaj način, osim ukoliko želite da vaš program ima samo jednu instrukciju!*

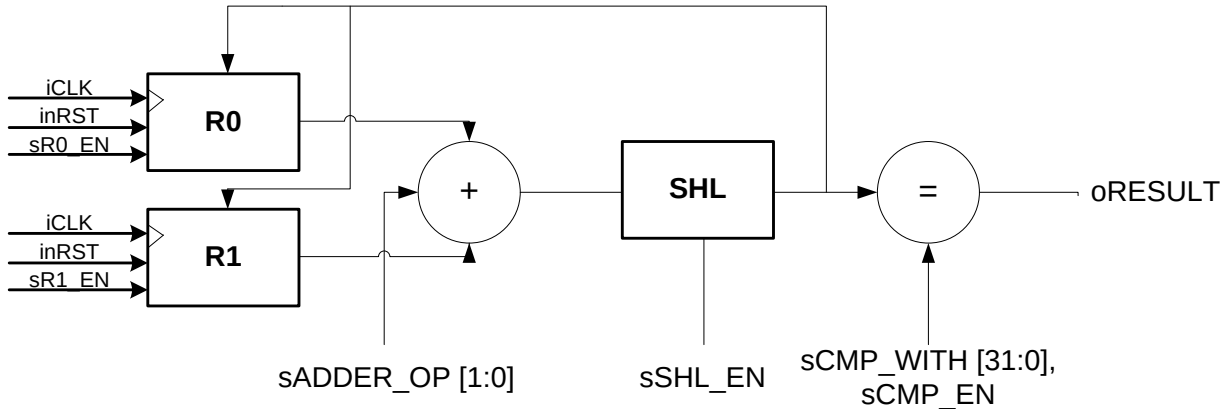
- **Upravljačke jedinice zasnovane na automatu sa konačnim brojem stanja** prolaze kroz niz stanja i time izvršavaju niz instrukcija. Trenutno stanje automata i, ukoliko postoji, trenutna instrukcija, određuju izlaz automata, tj. kontrolne signale ka ostatku sistema.
    - Opšte upravljačke jedinice se implementiraju na ovaj način ukoliko instrukciju treba izvršiti u više faza, gde svaka faza odgovara stanju automata
    - Specifične upravljačke jedinice se na ovaj način implementiraju veoma često, a tada jedno stanje automata odgovara jednoj instrukciji programa
- **Mikroprogramske upravljačke jedinice** izvršavaju mikroprogram definisan u mikroprogramskoj memoriji unutar upravljačke jedinice. Mikroprogram definiše niz kontrolnih signala potrebnih da bi se izvršila data instrukcija i/ili program.
    - Opšte upravljačke jedinice se implementiraju na ovaj način ukoliko instrukciju treba izvršiti u više faza, a svaka faza odgovara jednoj mikroinstrukciji mikroprogama unutar upravljačke jedinice
    - Specifične upravljačke jedinice se implementiraju na ovaj način ukoliko je željeni program zapamćen unutar upravljačke jedinice kao mikroprogram.

Na ovom predmetu nećemo implementirati mikroprogramske upravljačke jedinice, zbog njihove složenosti. U ovoj vežbi ćemo implementirati specifičnu upravljačku jedinicu i primer opšte kombinacione. U narednim vežbama imaćete priliku projektovati i opštu upravljačku jedinicu zasnovanu na automatu.

ZADACI

3. Aritmetički deo strukture za računanje

Kao prvi korak, implementiraćemo strukturu za računanje sa slike 3-1.



Slika 3-1. Primer aritmetičkog dela strukture za računanje

Opišite ovu upravljačku jedinicu u jednoj VHDL datoteci. Jedini ulazi sistema su takt i reset, dok je jedini izlaz oRESULT.

Svi kontrolni signali trebaju biti definisani kao *interni signali*. Oni će biti definisani upravljačkom jedinicom koju ćemo naknadno dodati.

Referencirajte se na teorijski deo vežbe za objašnjenje funkcionisanja svake od komponenata.

4. Specifična upravljačka jedinica

Sada ćemo implementirati jednostavnu specifičnu upravljačku jedinicu koja izvršava sledeći kratki program:

```
R0 <- (R0 + R1) << 1
R1 <- R1 + 1
R1 == 4 ?
```

Odlučite šta treba da budu vrednosti upravljačkih signala da bi se implementirala svaka od instrukcija ovog programa. Implementirajte upravljačku jedinicu kao automat sa 3 stanja, jedan po instrukciji. Neka se automat beskonačno vrti između ova tri stanja.

Početne vrednosti registara (u resetu) postavite na R0 = 0 i R1 = 0.

Ovaj program računa vrednost sledeće sume:

$$\sum_{i=0}^{N-1} i * 2^{N-i}$$

Kada oRESULT dobije vrednost 1, tj. kada je R1 = 4, vrednost registra R0 je vrednost ove sume za N = 4. Promenom vrednosti sa kojom se poredi vrednost registra R1 moguće je lako ovaj program modifikovati tako da računa vrednost sume za proizvoljno N (naravno, ograničeno širinom registra).

Opciono pokušajte modifikovati upravljačku jedinicu tako da kada vrednost oRESULT dostigne vrednost 1, automat prelazi u četvrto, konačno stanje, u kome sistem ne radi ništa (NOP) i automat ostaje stalno u tom stanju.

5. Opšta upravljačka jedinica implementirana kao kombinaciona mreža

Sada ćemo modifikovati upravljačku jedinicu tako da ona prima instrukcije. Ovom modifikacijom upravljačka jedinica postaje opšta i može da realizuje proizvoljan program u okviru skupa instrukcija koje podržava.

Dodaćemo još jedan ulaz sistema, iINSTR[4:0], koji će predstavljati 5-bitnu instrukciju. Instrukcija treba da bude poslata ka upravljačkoj jedinici.

Podržaćemo skup od šest instrukcija, prikazanih u tabeli 5-1.

Tabela 5-1. Instrukcije podržane strukturom za računanje

Mnemonik	Kod	Instrukcija
ADD	000	dest <= R0 + R1
INC	001	R1 <= R1 + 1
MOV	010	dest <= src
CMPZ	011	R1 == 0 ?
SHL	100	dest <= src << 1
ASH	101	dest <= (R0+R1) << 1

Instrukcija ima sledeći format:

- 3 bita – kod instrukcije
- 1 bit – odredišni registar (*dest*)
- 1 bit – izvorni registar (*src*)

Na primer, instrukcija 01001 je MOV instrukcija iz registra R1 u registar R0.

Implementirajte ovakvu upravljačku jedinicu podržavajući ovih šest instrukcija kroz kombinacionu mrežu – direktno definišući kontrolne signale u zavisnosti od trenutne instrukcije.

Simulirajte strukturu za računanje koristeći ove tri instrukcije iz prethodnog problema kao ulaze (definišite ih u test benchu).

ZAKLJUČAK

Čestitamo na implementaciji vaše prve strukture za računanje! Struktura za računanje koju ste upravo implementirali je veoma jednostavan primer procesora – strukture koja može da računa i odgovara na instrukcije. Ovaj procesor je veoma ograničen u svojim mogućnostima, ali sada ste spremni da projektujete konačni zadatak ovog predmeta – univerzalnu strukturu za računanje sa opštom upravljačkom jedinicom – RISC procesor.