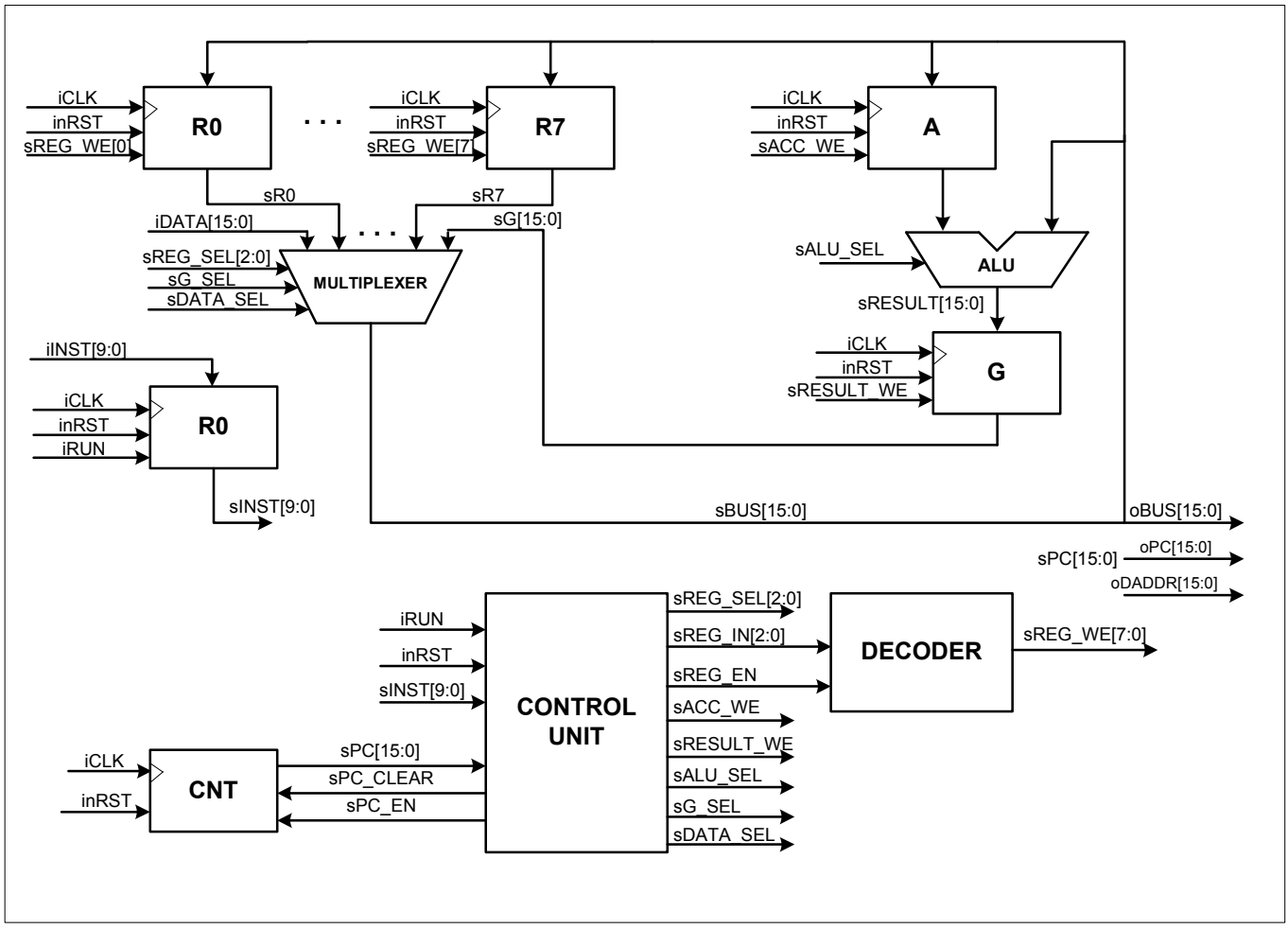
NAZIV VEŽBE:
Procesor.

ZADATAK VEŽBE:
Realizovati mikroalgoritam za računanje srednje vrednosti 8 brojeva

CILJ VEŽBE:
Upoznavanje se realizacijom procesora u VHDL-u na primeru projektovanja i simulacije zadatog mikroalgoritma.

1. Opis procesora

Slika 1.1 prikazuje blok dijagram realizovanog šesnaestobitnog procesora. Procesor sadrži deset 10 šesnaestobitnih registara, jedan devetobitni registar, multiplekser za selekciju sadržaja na procesorskoj magistrali, aritmetičku jedinicu, programski brojač, upravljačku jedinicu i dekoder za generisanje signala dozvole upisa u odredišni registar.



Slika 1.1 Blok dijagram realizovanog procesora (CPU_CORE.VHD)

Trenutna instrukcija `iINST[9:0]` se smešta u registar instrukcije kada je aktivan signal dozvole rada procesora označen sa `iRUN`.

Ulazni podaci se preko šesnaestobitnog ulaza `iDATA` i multipleksera upisuju u banku registara `R0-R7` i/ili u registar `A`. Izlaz multipleksera je šesnasetobitna magistrala podataka `sBUS`, koja je ujedno i ulaz registara `R0-R7`, `A` i izlaz iz digitalnog sistema koji opisuje ovaj procesor. Aritmetičko logička jedinica podržava samo operacije sabiranja i oduzimanja dva operanda. Ove operacije se izvode tako što se prvi operand smesti u registar `A` u jednom ciklusu takta, a drugi se u sledećem ciklusu dovede na magistralu `sBUS`. Rezultat operacija sabiranja i oduzimanja se smesta u registar `G`, registar za smeštanje rezultata, a nakon toga rezultat se može proslediti u neki od registara `R0-R7`, u registar `A` ili prema izlazu procesora prosleđivanjem na magistralu `sBUS`.

Upravljačka jedinica generiše selekcione i signale dozvole ostalih delova digitalnog sistema kao sto su: `sREG_SEL[2:0]` (odabir registra iz registarske banke za izlaz na procesorsku magistrali), `sREG_IN[2:0]` (kod registra u

koji će se upisati novi sadržaj), `sREG_EN` (dozvola upisa u adresirani registar), `sACC_WE` (dozvola upisa u akumulatorski registar), `sRESULT_WE` (dozvola upisa u registar rezultata ALJ), `sALU_SEL` (kod operacije aritmetičko logičke jedinice), `sPC_CLEAR` (brisanje sadržaja programskog brojača) i `sPC_EN` (dozvola povećanja programskog brojača) u zavisnosti od trenutnog koda instrukcije – `iINST[9:0]`.

Na osnovu koda registra `sREG_IN[2:0]` gde se upisuje nova vrednost i signala dozvole upisa `sREG_EN` koje generiše upravljačka jedinica procesorski dekoder definiše signale dozvole upisa u sve registre procesorske banke od 8 registara. Signal dozvole upisa u registarsku banku je označen sa `sREG_WE`.

Pored sadržaja procesorske magistrale `sBUS`, na izlazu procesora se obezbeđuje i trenutna vrednost programskog brojača `sPC` radi adresiranja trenutne instrukcije i vrednost adrese u memoriji podataka, `oDADDR`, odakle se učitava ulazni podatak preko magistrale `iDATA`.

Procesor treba da obavlja operacije koje prikazuje Tabela 1.1. U levoj koloni tabele su navedena imena instrukcija i oznake operanada, a u desnoj koloni funkcije pojedinih instrukcija. Sintaksa RX ←[RY] označava da se sadržaj registra Y upisuje u registar X. Instrukcija **mv** (engl. *move*) dozvoljava da se podaci kopiraju iz jednog registra u drugi. Instrukcija **mvi** (engl. *move immediate*) upisuje sadržaj sa ulazne magistrale podataka u registar X. To se označava sa RX ← D. U ovom primeru se neposredni operand iz koda instrukcije postavlja na ulaznu magistralu podataka `iDATA`, kombinovan sa vrednošću pročitanom iz memorije podataka.

Operacija	Funkcija	Kod instrukcije
`mv Rx,Ry`	RX ← [RY]	0000
`mvi Rx,Ry`	RX ← D	0001
`add Rx,Ry`	RX ← [RX] + [RY]	0010
`sub Rx,Ry`	RX ← [RX] - [RY]	0011

Tabela 1.1 Intrukcije procesora

Pored instrukcija upisivanja vrednosti u registar, procesor podržava i instrukcije za sabiranje (**add**) i oduzimanje (**sub**) sadržaja dva registra RX i RY. Pri tome, rezultat se uvek smešta u prvi navedeni registar RX.

Svaka instrukcija je kodovana sa 10 bita. Lista instrukcija se smešta u posebnu memoriju, odvojenu od memorije za smeštanje podataka. Format instrukcije na bitskom nivou je IIIIXXXYYY. Pri tome je sa IIII označen kod intrukcije, sa XXX adresa registra RX a YYY adresa registra RY.

Iako su za navedene instrukcije dovoljna dva bita, ovde se rezervišu četiri iz razloga proširenja skupa instrukcija koji će se raditi u ovoj i narednoj vežbi.

Izvršenje svih instrukcija je raspoređeno u četiri faze (četiri periode takta):

1. prihvat insntrukcije,
2. dekodovanje instrukcije,
3. izvršenje instrukcije i
4. određivanje naredne adrese.

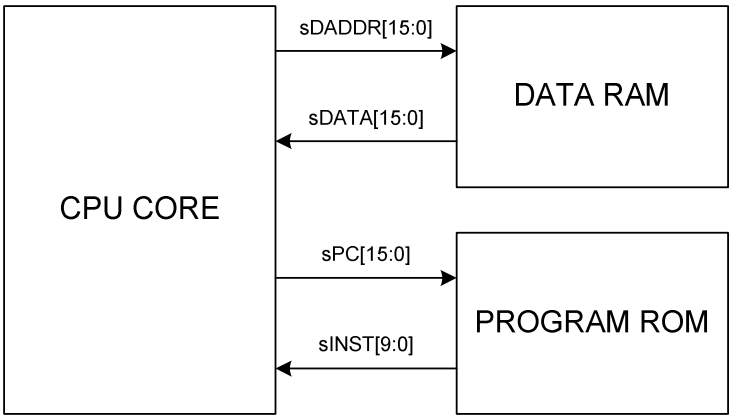
Ovakvo raspoređivanje izvršenja instrukcija je potrebno jer neke od njih zahtevaju više od jednog prenosa podataka između registara radi izvršenja instrukcije. Na primer, prilikom sabiranja sadržaja registara RX i RY, prvo je potrebno sadržaj registra RX upisati u registar A, zatim RY pustiti na procesorsku magistrali i sadržaj aritmetičke operacije upisati u registra G nakon čega sledi upis rezultat u odredišni registar. Ilustraciju toka podataka kroz procesor i generisanje signala dozvole upisa u registar prikazuje Tabela 1.2.

	T_0 PRIHVAT	T_1 DEKODOVANJE	T_2 IZVRŠENJE	T_3 NAR. INSTR.
I0: `mv`		RX in, RY out	PC EN	RX WE
I1: `mvi`		RX in, D out	PC EN	RX WE
I2: `add`		RX out, A in, ALU sel	RY out, G WE, PC EN	G sel, RX WE
I3: `sub`		RX out, A in, ALU sel	RY out, G WE, PC EN	G sel, RX WE

Tabela 1.2 Ilustracija generisanja upravljačkih signala za izvšenje instrukcija

Pošto je upravljačka jedinica procesora realizovana sa registrovanim izlazima, na vremenskom dijagramu simulacije rada procesora svi upravljački signali kasne za jednu periodu takt signala u odnosu na nevedeni redosled koji prikazuje Tabela 1.2.

Jezgro procesora je povezano sa dve memorije: memorija sa instrukcijama i memorija sa podacima. Blok šemu povezivanja prikazuje Slika 1.2. Za adresiranje memorije sa podacima iskorišten je registar R0.



Slika 1.2. Blok šema povezivanja procesorskog jezgra i memorije (CPU.VHD)

2. Zadatak

Za definisani procesor isprojektovati mikroalgoritam računanja srednje vrednosti osam brojeva unetih kao neposredni operandi. Radi realizacije ovog cilja, potrebno je proširiti skup instrukcija datog procesora sa dve nove instrukcije koje prikazuje Tabela 2.1.

Operacija	Funkcija	Kod instrukcije
`shl Rx,Ry`	RX ← shl[RY]	0100
`shr Rx,Ry`	RX ← shr[RY]	0101

Tabela 2.1. Nove instrukcije procesora

Operacije pomeranja realizovati proširenjem skupa aritmetičkih operacija koje izvršava ALJ (CPU_ALU.VHD) sa dve nove instrukcije koje će realizovati pomeranje prvog operanda ALJ (operand označen sa A) za jedno mesto ulevo u slučaju operacije shl odnosno udesno za operaciju shr. Pored toga potrebno je i proširiti upravljačku jedinicu procesora sa dekodovanjem novih instrukcija.

U okviru vežbe je potrebno :

- Formirati blok dijagram algoritma za računanje srednje vrednosti osam brojeva
- Formirati mikrokod na osnovu definisanog blok dijagrama
- Formirati tabelu generisanja upravljačkih signala za nove instrukcije
- Izmeniti VHDL kod sa ciljem izvršavanja projektovanog mikrokoda
- Simulirati izvršenje mikroalgoritma pomoću testbencha koji sam proverava tačnost rezultata
  Testbench treba da simulira rad modula opisanog u datoteci CPU.VHD
- Izvršiti sintezu procesora za TLL5000 demonstracionu ploču, pomoću datog VHDL modula za povezivanje sa potrebnim periferijama na TLL5000 ploči (CPU_SHELL.VHD)
- Izvršiti testiranje u realnom vremenu na TLL5000 demonstracionoj ploči

Niz od 8 brojeva treba smestiti u memoriju sa podacima (entitet `CPU_DATA_ROM`).

REALIZACIJA VEŽBE:

U okviru vežbe je potrebno kreirati projekat za realizaciju traženog digitalnog sistema i izvršiti njegovu simulaciju uz pomoć VHDL testbench-a.

PRIPREMA ZA VEŽBU:

Priprema za vežbu obuhvata sledeće:

- Formirati tabelu generisanja upravljačkih signala za nove instrukcije
- Izmeniti VHDL kod procesora dodavanjem dve nove instrukcije za pomeranje ulevo i udesno sadržaja registra
- Formirati blok dijagram algoritma za računanje srednje vrednosti osam brojeva
- Formirati mikrokod na osnovu definisanog blok dijagrama

DODATNI ZADATAK:

1. Izmeniti VHDL kod tako da procesor sadrži statusni registar sa četiri statusna bita Z, N, O i C.
   Stanje četiri statusna bita definisati na osnovu sadržaja registra `G` i prikazati na LED diodama
2. Formirati mikroalgoritam koji redom pali i gasi LED diode