|  |  |
| --- | --- |
| title-left | **ELEKTROTEHNIČKI FAKULTET**  BEOGRAD |

**Računarski VLSI sistemi**

student: Milan Branković

119/07

profesor: dr Veljko Milutinović

asistent: Saša Stojanović

|  |  |
| --- | --- |
| broj osvojenih poena: |  |

Beograd

10.01.2011

**DVOADRESNI PROCESOR SA SINHRONOM MAGISTRALOM**

Milan Branković

*email: milan.brankovic@rocketmail.com*

# 1. Definisanje projekta

## 1.1 Uvod

Projekat je urađen kao domaći zadatak iz predmeta *Računarski VLSI sistemi* na 4. godini studija IR odseka, Elektrotehničkog fakulteta Univerziteta u Beogradu i namenjen je samo u edukativne svrhe.

Ovaj rad predstavlja implementaciju projekta iz predmeta Arhitektura i organizacija računara 2, profesora Jovana Đorđevića.

Projekat predstavlja dvoadresni 16-to bitni procesor. Realizovan je u alatu Quartus 9.1, kompanije Altera i testirano je sintentizovanje modela na Cyclon II familiji čipova.

Svrha i važnost ovog projekta je realizacija potpuno funkcionalne hardverske jedinice u VHDL-u.

## 1.2 Ciljevi projekta

Cilj projekta je upoznavanje sa VHDL-om, jezikom za opis hardvera, kao i sticanje praktičnih iskustava u radu sa alatima za prevođenje, simulaciju i sintezu sistema opisanih hardverskim jezikom.

Smatram da su implementacija ovoga u VHDL-u, a i kasnija implementacija u hardveru veoma interesantne i korisne za dalje projekte. Istina da je ovaj procesor skromnih mogućnosti i da sam za sebe predstavlja samo jezgro nekog većeg sistema, ali smatram i da je lako nadograditi ga, poboljšati mu performanse i ugraditi ga u bilo koju složeniju aplikaciju.

# 2. SpecifikacijA

## 2.1 Uvod

Sistem se sastoji od procesora i memorije koji su povezani adresnom magistralom, magistralom podataka, i kontrolnom signalima za operaciju iniciranja čitanja odnosno upisa u memoriju.

## 2.2 Spoljašnji interfejsi

Procesor ima 35 pinova koji se koriste za adrese, podatke i kontrolne signale. Pinovi se vezuju na 1 adresnu magistralu magistralu, 1 magistralu podataka i kontrolne signale. Procesor podržava mehanizam prekida.

Magistrala ima 28 pinova koji se koriste za adrese, podatke i kontrolne signale. Pinovi se vezuju na 1 adresnu magistralu magistralu, 1 magistralu podataka i kontrolne signale.

Procesor i magistrala su objedinjeni u jedinstvenu komponentu.

Komunikacija između procesora i memorije se obavlja na sledeći način:

* *Slučaj čitanja podatka iz memorije*: procesor na adresnu magistralu postavlja adresu memorijske lokcaije sa koje želi da pročita podatak, i neaktivnom vrednošću kontrolnog signala RDBUS signalizira memoriji da želi da pročita podatak. Memorija odgovara tako što postavi podatak na linije podataka, koji procesor očita.
* *Slučaj upisa podatka u memoriju*: procesor na adresnu magistralu postavlja adresu memorijske lokacije na koju želi da upiše podatak koji postavlja na magistralu podataka, i neaktivnom vrednošću kontrolnog signala WRBUS signalizira memoriji da želi da upiše podatak.

## 2.3 Specifikacija hardvera

Od hardverskih komponenti korišćene su sledeće:

- D flip-flop, SR flip-flop

- koder

- dekoder

- multiplekser

- 16-bitni registar

- 16-bitni registar sa mogućnošću inkrementiranja i dekrementiranja

- 8-bitni registar sa mogućnošću inkrementiranja i dekrementiranja

## 2.4 Specifikacija softvera

Za izradu projekta korišćene su sledeće biblioteke:

IEEE.STD\_LOGIC\_1164.ALL

IEEE.STD\_LOGIC\_SIGNED.ALL

# 3. Unutrašnji blokovi

Procesor je podeljen na pet blokova: blok registri, blok operacije, blok prekidi, blok interfejs koji zajedno predstavljaju operacionu jedinicu, i upravljačka jedinica.

Svaki od ovih podblokova je podeljen na manje podblokove radi lakšeg snalaženja korisnika.

***Blok interfejs*** sadrži 16-tobitni registar MAR koji služi za čuvanje adrese koja će se koristiti pri ciklusu upisa ili čitanja. Pored njega ovde se nalazi (-bitni registar MBR u kome se čuva podatak koji je očitan ili koji treba upisati u memoriju. Takođe u ovom bloku se nalazi mreža za arbitraciju i sinhronizaciju na magistrali.

***Blok registri*** sadrži registarski fajl (registe AX, BX, CX, DX, SI , DI, SP, BP), programski brojač PC, pomoćne registre A i B, četvoro bajtni instrukcijski registar IR , kao i mrežu za dekododovanje operacija i načina adresiranja, statusni registar PSW, i mrežu za selekciju registara.

***Blok operacije*** se sastoji od dva pomoćna registra X i Y, i aritmetičko logičke jedinice ALU.

***Blok prekidi*** se sastoji od flip-flopova koji služe za pamćenje vrste prekida, mreže za prioritiranje prekida, registra ukazivača na početak tabele prekidnih rutina IVTP, pomoćnog registra BR i trobitnog registra maske IMR.

***Upravljačka jedinica*** se sastoji od dva podbloka. U prvom se nalazi mreža za generisanje i upis nove vrednosti u brojač koraka i brojač koraka, dok se drugi podblok sastoji od dve komponente od kojih je jedna odgovorna za generisanje upravljačkih signala operacione jedinice, a druga je odgovorna za generisanje upravljačkih signala upravljačke jedinice.

# 4. Opis dizajna

## 4.1 Zabeleške uz dizajn

Za sinhronizaciju između procesora i memorije usvojena je sledeća predpostavka: inicijalna vrednost brojača za sinhronizaciju unutar memorije je za jedan manja od inicijalne vrednosti brojača za sinhronizaciju procesora.

U nastavku su date tabele načina kodovanja operacija i načina adresiranja.

|  |  |
| --- | --- |
| Code | Operation |
| 0000 0001 | RTS |
| 0000 0010 | RTI |
| 0000 0100 | INTE |
| 0000 1000 | INTD |
| 0001 0000 | TRPE |
| 0010 0000 | TRPD |
| 0100 0000 | JMP |
| 0101 1111 | JSR |
| 0110 0000 | BNZ |
| 0111 1111 | INT |
| 1000 0001 | MOVS |
| 1000 0010 | MOVD |
| 1000 0100 | ADD |
| 1000 1000 | AND |
| 1000 0000 xx00 0xxx | ASR |
| 1111 1111 xx00 0xxx | PUSH |
| 1111 1111 xx00 1xxx | POP |
| 1111 1111 xx01 0xxx | INC |
| 1111 1111 xx01 1xxx | DEC |
| 1111 1111 xx10 0xxx | JMPIND |

*Tabela 1. Spisak kodova operacija i njihovo značenje*

|  |  |
| --- | --- |
| Code | Way of addressing |
| 00xx xZZZ | Registarsko direktno  [AX, BX, CX, DX, SI, DI, SP, BP] |
| 01xx xZZZ | Registarsko indirektno  [BX, SI, DI, BP, BX+SI, BX+DI, BP+SI, BP+DI] |
| 10xx xZZZ | Registarsko indirektno sa pomerajem [BX, SI, DI, BP, BX+SI, BX+DI, BP+SI, BP+DI] |
| 11xx xx00 | Memorijsko direktno |
| 11xx xx01 | Memorijsko indirektno |
| 11xx xx10 | Relativno |
| 11xx xx11 | neposredno |

*Tabela 2. Spisak kodova načina adresiranja i njihovo značenje*

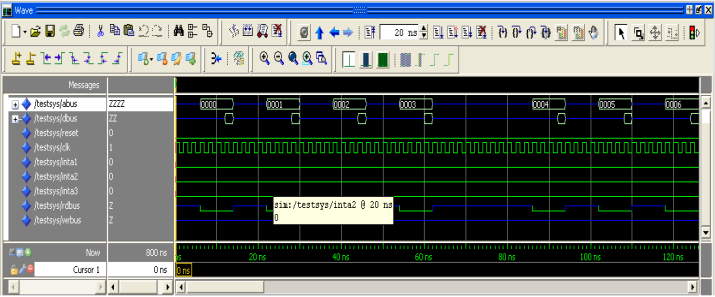
Legenda:

Kodovi operacija dati su prvim i drugim bajtom instrukcije, dok su kodovi za načine adresiranja dati drugim bajtom instrukcije, pri čemu su usvojena sledeća značenja za simbole:

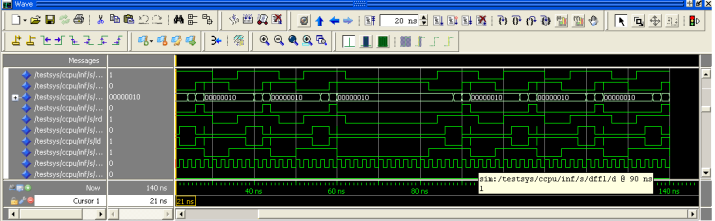
x – nebitno

ZZZ – 0 .. 7 odnosi se na registre u zagradama navedenim uz način adresiranja.

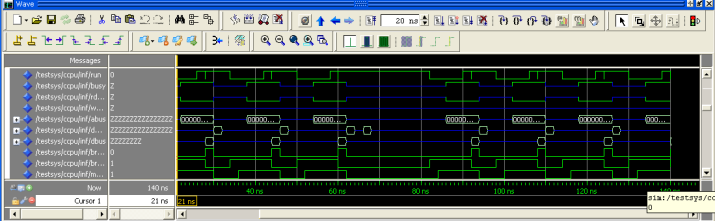
## 4.2 Vremenski dijagrami



*Slika 1. Dijagram vremenskih signala test primera*



*Slika 2. Dijagram vremenskih signala*



*Slika 3. Dijagram vremenskih signala pri sinhronizaciji*

## 4.3 Konačni automati



*Slika 4. Dijagram stanja procesora*

Procesor prolazi kroz sledeće stanja : IF, ID, EX i HALT.

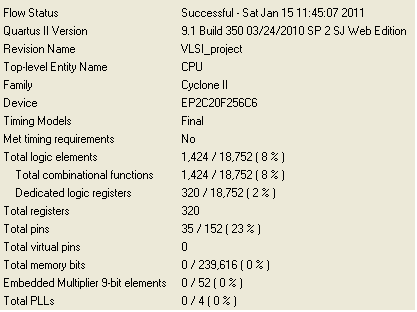
IF (Instruction fetch) faza odnosno stanje je stanje u kome se procesor nalazi na početku izvršavanja svake instrukcije. U ovoj fazi se određuje dužina instrukcije, i iz memorije se učitava onoliko bajtova kolika je dužina instrukcije, a to je određeno prvim bajtom instrukcije.

ID (Instruction Decode) stanje je stanje u kome procesor određuje koji način adresiranja se koristi u instrukciji, i o kojoj instrukciji se radi. Određivanje načina adresiranja i određivanje instrukcije se vrši uz pomoć instrukcijskog registra.

EX (Execution) stanje je stanje u kome procesor izvršava instrukciju, i proverava da li je došlo do prekida, kao i smeštanje programskog brojača i programske statusne reči na stek, i čitanje novih radi izvršavanje instrukcija prekidne rutine ukoliko je došlo do prekida.

HALT stanje je stanje u kome je procesor kada završi izvršavanje svih instrukcija tj. celokupnog programa .

## 4.4 Tabele



*Slika 5. Tabela sintetizovanog modela*

# 5. Testiranje i verifikACIJA

## 5.1 Simulacija i test benčevi

Quartus ne podržava standardne vhdl testbench-ove tako da je umesto toga korišćen ModelSim za testiranje i verifikaciju.

Za potrebe simulacije izvršene su simulacije pojedinačnih komponenti, kao i celokupnog sistema. Za testiranje i simulaciju celokupnog sistema napravljen je poseban entitet, koji kontroliše rad procesora, puni memoriju instrukcijama koje će procesor izvršavati, i generiše signal takta.

MOVS cx, #5  
MOVS bx, #100  
MOVS ax, #1  
MOVD ax, [bx]  
INC ax  
INC bx  
INC bx  
DEC cx  
BNZ -12  
MOVS cx, #5  
MOVS bx, #100  
MOVS ax, #0  
MOVS dx, [bx]  
ADD ax, dx  
INC bx  
INC bx  
DEC cx  
BNZ -12  
MOVD ax, [bx]

*Slika 6.Test bench*

# 6. ImplementacijA

Implementacija je urađena za čip EP2C5AF256A7, familije Cyclone II. Dizajn je urađen u programskom alatu Quatrus II verzija 9.1, a za sumulaciju je korišćen alat ModelSim verzije 6.5b, oba firme Altera.

# 7. Revizije i komentari

Prilikom izrade projekta konsultovao sam kolege sa kojima sam radio projekat iz predmeta Arhitektura i organizacija računara 2.

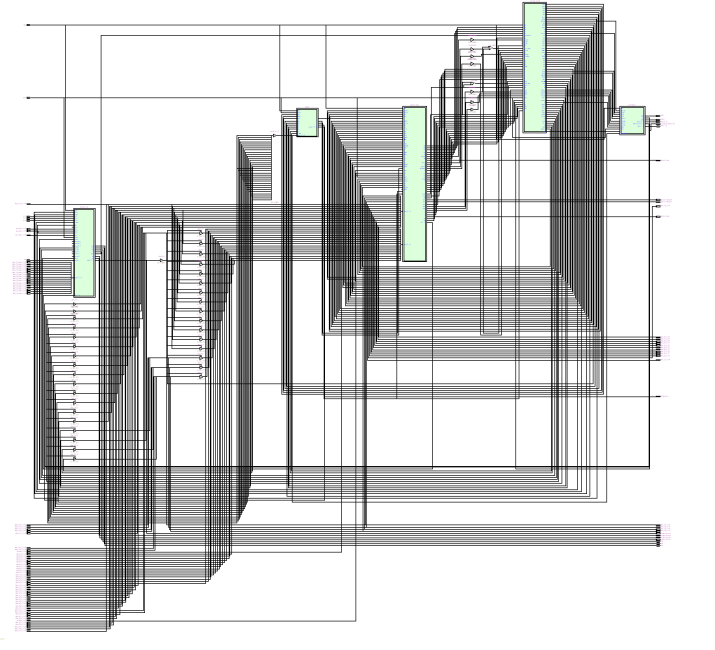
# 8. Literatura

Spisak literature korišćen u izradi domaćeg zadatka u IEEE formatu, npr:

|  |  |
| --- | --- |
| [1] | Saša Stojanović, *Skripta iz predmeta VLSI ,* ETF, Beograd, 2010. |
| [2] | <http://www.altera.com/>, USA |
| [3] | <http://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html> |

# 9. Prilozi

**9.1 Sintetizovani prikaz procesora**



*Slika 7. Sintetizovan prikaz procesora*

**9.2 Spisak fajlova izvornog koda**

Alu.vhd

Arbit.vhd

cd.vhd

cnt.vhd

cntrl\_unit.vhd

cntrl\_unit1.vhd

cntrl\_unit2.vhd

CPU.vhd

D\_FF.vhd

dc.vhd

gen\_ld\_out.vhd

inc\_dec\_reg.vhd

interface.vhd

interupts.vhd

ir.vhd

ivtp\_br.vhd

mar\_mbr.vhd

memory.vhd

mx2to1.vhd

mx4to1.vhd

pc.vhd

prio.vhd

psw.vhd

reg.vhd

reg\_file.vhd

reg\_sel.vhd

registers.vhd

regPC\_A\_B.vhd

sinch.vhd

sp.vhd

SR\_FF.vhd

testSYS.vhd

txt\_util.vhd