Дизајн микропроцесора риск арихтектуре

Архитектура микропроцесора

- 16 регистара опсте намене (16-битни)
- 16 адресних линија за комуникацију са меморијом
- 16 линија за податке за читање/упис података из/у меморију
- 14 инструкција

Скуп инструкција

Инструкција	Операција	Сематика операције
ADD d,s0,s1	сабирање	d←s0+s1
INC d,s0	инкрементирање	d←s0++
SUB d,s0,s1	одузимање	d←s0-s1
NOT d,s0	битска негација	d←~s0
AND d,s0,s1	битска коњукција	d←s0&s1
OR d,s0,s1	битска дисјункција	d←s0|s1
SLA d,s0	померање улево за 1	d←s0<<1
SRA d,s0	померање удесно за 1	d←s0>>1
LD d,s0	учитавање из меморије	d←(s0)

Скуп инструкција

ST d,s0	упис у меморију	(d)←s0
LDI d,imm	учитавање броја	d←s0+s1
JMP imm	безусловни скок	goto PC+imm
JZ imm	условни скок (ако је једнако)	If(Z) goto PC+imm
JB imm	условни скок (ако је мање)	If(!Z && C) goto PC+imm

- d, s0, s1 су неки од регистара опште намене,
- знак ← представља упис у неки од регистра или у меморију,
- заграде () представљају да регистар садржи адресу неке меморијске локације,
- imm је 8-битни означен број који је задат у инструкцији,
- PC (program counter) регистар у коме се чува адреса текуће инструкције,
- Z i C su zero i carry flegovi.

Организација микропроцесора

Основне компоненте микропроцесора су:

- аритметичко-логичка јединица
- контролна јединица
- скуп регистара
- меморијска јединица
- улазно-излазна јединица

Опис модела разматраног система дат је у Verilog HDL језику.

Модел аритметичко-логичке јединице

```
26 `define ALU_ADD 3'b000
27 `define ALU_INC 3'b001
28 `define ALU_SUB 3'b010
29 `define ALU_NOT 3'b011
30 `define ALU_AND 3'b100
31 `define ALU_OR  3'b101
32 `define ALU_SLA 3'b110
33 `define ALU_SRA 3'b111
```

Макрои убрзавају писање *HDL* описа и повећавају његову прегледност.

Дефинисање макро коришћених у *ALU* модул.

```
85 ////////////////////////////////////////////////////////
86 //////// MODEL ARITMETICKO-LOGICKE JEDINICE ////////////
87 ////////////////////////////////////////////////////////
88 module alu (
89 input ena,              // Aktiviranje komponente.
90 input [15:0] in0,       // Prvi ulazni operand.
91 input [15:0] in1,       // Drugi ulazini operand.
92 input [2:0] op,         // Kod operacije.
93 output reg [15:0] out, // Izlazni port(rezulatat operacije).
94 output zero,        // Zero fleg.
95 output reg carry);     // Carry fleg.
```

Модул је основни градивни елемент који чини основу пројектовања у *Verilog* језику, а портови представљају спрежне тачке преко којих модул међусобно комуницирају.

Дефинисање *ALU* модула и његове листе портова.

Модел аритметичко-логичке јединице

```
 97 always @(ena or in0 or in1 or op)
 98    if(ena == 1)
 99       case(op)
100          `ALU_ADD: {carry, out} <= in0 + in1;
101
102          `ALU_INC: {carry, out} <= in0 + 16'd1;
103
104          `ALU_SUB: {carry, out} <= in0 - in1;
105
106          `ALU_NOT: {carry, out} <= {1'b0,~in0};
107
108          `ALU_AND: {carry, out} <= {1'b0,in0&in1};
109
110          `ALU_OR: {carry, out} <= {1'b0,in0|in1};
111
112          `ALU_SLA: {carry, out} <= {in0,1'b0};
113
114          `ALU_SRA: {out, carry} <= {in0[15], 1'b0};
115       endcase
116       assign zero = {out == 16'd0};
117 endmodule
```

Главни део кода *ALU* модула где се користи процедура типа *always,* условна додела *if*, структура *case*, континуална додела *assign.* Знак <= представља неблокирајуће доделе.

Always процедура се извршавати кад гок дође до промене неког од *ena, in0, in1* ili *op* сигнала.

Модул скупа регистара опште намене

```
////////////////////////////////////////////////////////////////
//////// MODEL SKUPA REGISTARA OPSTE NAMENE ////////////////////
////////////////////////////////////////////////////////////////
module regfile(
input [3:0] sel_out0, // Signal selekcije registra cija ce vrednost biti predstavljena na izlazu out0.
input [3:0] sel_out1, // Signal selekcije registra cija ce vrednost biti predstavljena na izlazu out1.
input ena_in,         // Kontrolni registar za upis u registar.
input [3:0] sel_in,   // Signal selekcije registra u koji ce biti upisana vrednost sa ulaza in.
input [15:0] in,      // Vrednost koja se upisuje u selektovani registar.
output [15:0] out0,   // Vrednost selektovanog registra sa portom sel_out0.
output [15:0] out1);  // Vrednost selektovanog registra sa portom sel_out1.

reg [15:0] regs[15:0]; // Skup registra opste namene mikroporocesora.

assign out0 = regs[sel_out0];
assign out1 = regs[sel_out1];

always @(ena_in or sel_in or  in)
   if(ena_in)
      regs[sel_in] <= in;
endmodule
```

Модел *datapath* компоненте

- Ова компонента врши пренос података унутар микрокопроцесора.
- Обједињује импмплементацију магистрале, наменских регистара и рудиментарне меморије.
- Магистрала представља два супа од по 16 жица.
- Наменски регистри који се налазе у овој компоненти су:
  - MAR(memory address register)
  - MDR(memory data register)
  - PC(program counter)
  - IR(instruction register)
  - PSW(Program Status Word)

Модел *datapath* компоненте

Листа улазних портова и њихова декларација.

```
148 module datapath(
149 input clk,                    // Signal takta.
150 input rst,                    // Signal reseta.
151 input mem_op,                 // Odredjuje pristup memoriji (upus ili citanje).
152 input [1:0] a_sel,            // Odredjuje sta se pusta na magistralu a.
153 input [1:0] b_sel,            // Odredjuje sta se pusta na magistralu a.
154 input [3:0] regfile_out0_sel, // Selektovanje izlaznog out0 registra iz skupa registara.
155 input [3:0] regfile_out1_sel, // Selektovanje izlaznog out0 registra iz skupa registara.
156 input regfile_in_ena,         // Signal dozvole upisa u registra iz skupa registara.
157 input [3:0] regfile_in_sel,   // Selektovanje registra iz skupa registra u koji ce biti upusana vrednost.
158 input alu_ena,               // Aktiviranje aritmeticko-logicke jedinice.
159 input [2:0] alu_op,          // Kod operacije za alu jedinicu.
160 input pc_ena,                // Dozvola upisa PC registar.
161 input pc_sel,                // Selektovanje ulaza u PC registar.
162 input ir_ena,                // Dozvola upisa u IR registar.
163 input psw_ena,               // Dozvola upisa u PSW registar.
164 input mar_ena,               // Dozvola upisa u MAR registar.
165 input mdr_ena,               // Dozvola upisa u MDR registar.
166 input mdr_sel,               // Selektovanje ulaza u MDR registar.
```

Модел *datapath* компоненте

Листа излазних портова и њихова делкарација.

data порт је бидирекциони порт.

```
167  output [3:0] opcode,          // Kod operacije tekuce instrukcije.
168  output [3:0] dst,             // Adresa odredisnog registra tekuce instrukcije.
169  output [3:0] src0,             // Adresa prvog izvrsnog registra tekuce instrukcije.
170  output [3:0] src1,             // Adresa drugog izvrsnog registra tekuce instrukcije.
171  output zero,                   // Vrednost Z flega PSW registra.
172  output carry,                  // Vrednost C flega PSW registra.
173  output [15:0] addr,            // Memorijska adresa.
174  inout [15:0] data);            // Podatak koji se razmenjuje sa memorijom.
```

Декларација носиоца податка.

```
176  wire [15:0] bus_a, bus_b;      // Magistrale mikroprocesora.
177  reg [15:0] pc, ir;             // Registri PC i IR.
178  reg [1:0] psw;                 // Registar PSW.
179  reg [15:0] mar, mdr;           // Registri MAR i MDR.
180  wire [15:0] regfile_out0, regfile_out1; // Izlazi skupa registara opste namene.
181  wire [15:0] alu_res;           // Rezultat racunanja u ALU komponente.
182  wire alu_zero, alu_carry;      // Flegovi ALU komponente.
```

Модел *datapath* компоненте

Селектовање сигнала који се прослеђују на магистрале A и B.

```
184 assign bus_a = a_sel[1] ? pc : (a_sel[0] ? mdr : regfile_out0);
185 assign bus_b = b_sel[1] ? (b_sel[0] ? alu_res : {{8{ir[7]}},
186 ir[7:0]}):(b_sel[0] ? mdr : regfile_out1);
```

Разбијање неких регистара на чланове ради сталне њихове доступности када год се неки промени. За то разбијање се користи операција придруживања. На порт *data* се прослеђују вредност *MDR* регистра само ако је у питању упис у меморију, а на порт *addr* се прослеђује вредност регистра *MAR* када год се догоди његова промена.

```
209 assign {opcode, dst, src0, src1} = ir;
210 assign {carry, zero} = psw;
211 assign addr = mar;
212 assign data = mem_op ? mdr : 16'dz;
```

Модел *datapath* компоненте

```
188 always @(pc_ena or pc_sel or bus_b)
189    if(pc_ena)
190       if(pc_sel)
191          pc <= bus_b;
192       else
193          pc <= 16'd0;
194 always @(ir_ena or bus_a)
195    if(ir_ena)
196       ir <= bus_a;
197 always @(psw_ena or alu_zero or alu_carry)
198    if(psw_ena)
199       psw = {alu_carry,alu_zero};
200 always @(mar_ena or bus_a)
201    if(mar_ena)
202       mar <= bus_a;
203 always @(mdr_ena or mdr_sel or bus_b or data)
204    if(mdr_ena)
205       case(mdr_sel)
206          1'b0: mdr <= data;
207          1'b1: mdr <= bus_b;
208       endcase
```

Креирање процеса који контролишу упис у *PC, IR, PSW, MAR, MDR* регистре.

Упис у *PC i MDR* регистре се контролише мултиплексорима јер постоје два извора са којих долази вредност за упис у ове регистре.

Модел *datapath* компоненте

У овој компоненти се инстанцирају објекти модела скупа опште намене и компонента аритметичко-логичке јединице. Постоје два начина повезивања портова инстанцираних модела са сигналима из окружења:

- повезување преко уређене листе,
- повезивање поротва на основу њихових имена.

Овде је употребљена други начин повезивања.

```
214  /* Instanciranje skupa registara opste namene i aritmeticko-logicke jedinice*/
215  regfile _regfile(.sel_out0(regfile_out0_sel), .sel_out1(regfile_out1_sel),
216  .ena_in(regfile_in_ena), .sel_in(regfile_in_sel), .in(bus_b),
217  | .out0(regfile_out0), .out1(regfile_out1));
218
219  alu _alu(.ena(alu_ena), .in0(bus_a), .in1(bus_b), .op(alu_op),
220  .out(alu_res), .zero(alu_zero), .carry(alu_carry));
221  endmodule
```

Модел контролне јединице

Дефинисање макроа коришћених у моделу контролне јединице.

```
35  /* Koraci pri izvrsenju instrukcija. */
36  `define FETCH   2'b00
37  `define DECODE  2'b01
38  `define EXECUTE 2'b10
39  `define WRITE   2'b11
```

```
62  /* Operacije sa memorijom. */
63  `define MEM_READ  1'b0
64  `define MEM_WRITE 1'b1
```

```
66  /* Selektroski signali za magistralu A. */
67  `define BUS_A_SEL_REGFILE_OUT0 2'b00
68  `define BUS_A_SEL_MDR          2'b01
69  `define BUS_A_SEL_PC           2'b10
```

```
42  /* Kodovi pojedinacnih instrukcija. */
43  `define ADD 4'b0000
44  `define INC 4'b0001
45  `define SUB 4'b0010
46  `define NOT 4'b0011
47  `define AND 4'b0100
48  `define OR  4'b0101
49  `define SLA 4'b0110
50  `define SRA 4'b0111
51  `define LD  4'b1000
52  `define ST  4'b1001
53  `define LDI 4'b1010
54  `define JMP 4'b1011
55  `define JZ  4'b1100
56  `define JB  4'b1101
```

Модел контролне јединице

```
71  /* Selektroski signali za magistralu B. */
72  `define BUS_B_SEL_REGFILE_OUT1 2'b00
73  `define BUS_B_SEL_MDR          2'b01
74  `define BUS_B_SEL_IMM          2'b10
75  `define BUS_B_SEL_ALU_RES      2'b11
```

```
77  /* Selektorski signali za PC registar. */
78  `define PC_SEL_0     1'b0
79  `define PC_SEL_BUS_B 1'b1
```

```
81  /* Selktorski signali za MDR registar. */
82  `define MDR_SEL_DATA  1'b0
83  `define MDR_SEL_BUS_B 1'b1
```

```
58  /* Konstante koje ativiraju tj. deaktiviraju odredjene signale. */
59  `define ENABLE  1'b1
60  `define DISABLE 1'b0
```

Модел контролне јединице

Контролна јединица се понаша као коначни аутомат са 4 стања, где стања представљају одговарајући корачи при извршењу инструкције. Кораци при извршењу инструкције су, доношење, декодирање, извршење и упис.

Листа улазних портова и њихова делкарација. Поред сигнала такта и ресета улазни сигнали су и сигнали од *datapath* компоненте који су потребни за генерисање контролних сигнала.

```
223 ////////////////////////////////////////////////////////////
224 /////////////// MODEL KONTROLNE JEDINICE //////////////////
225 ////////////////////////////////////////////////////////////
226 module control (
227 input clk,                  // Signal takta.
228 input rst,                  // Reset signal.
229 input [3:0] opcode,         // Komponenta tekuce instrukcije: opkod.
230 input [3:0] dst,            // Komponenta tekuce instrukcije: adresa odredisnog registra.
231 input [3:0] src0,           // Komponenta tekuce instrukcije: adresa izvornog registra 0.
232 input [3:0] src1,           // Komponenta tekuce instrukcije: adresa izvornog registra 1.
233 input zero,                 // Vrednost zero flega psw registra.
234 input carry,                // Vrednost carry flega psw registra.
```

Модел контролне јединице

Листа излазних портова и њихова делкарација. Сви излазни сигнали представљају контролне сигнале који се повезују на одговарајуће улазе *datapath* јединице.

```
235  output reg mem_op,                  // Tip pristupa memoriji (upis ili citanje).
236  output reg [1:0] a_sel,             // Odredjivanje sta se propusta na a magistralu.
237  output reg [1:0] b_sel,             // Odredjivanje sta se propusta na b magistralu.
238  output reg [3:0] regfile_out0_sel, // Selektroski signal za out0 izlaz skupa registara.
239  output reg [3:0] regfile_out1_sel, // Selektroski signal za out1 izlaz skupa registara.
240  output reg regfile_in_ena,        // Kontrolni signal upisa u registar iz skupa registara.
241  output reg [3:0] regfile_in_sel, //Registra iz skupa registra u koji ce biti upisana vrednost.
242  output reg alu_ena,               // Ukljucenje aritmeticko-logicke jedinice.
243  output reg [2:0] alu_op,          // Kod operacije za aritmeticko-logicku jedinicu.
244  output reg pc_ena,                // Kontrolni signal za PC registar.
245  output reg pc_sel,                // Selektroski signal za PC registar.
246  output reg ir_ena,                // Kontrolni signal za IR registar.
247  output reg psw_ena,               // Kontrolni signal za PSW registar.
248  output reg mar_ena,               // Kontrolni signal za MAR registar.
249  output reg mdr_ena,               // Kontrolni signal za MDR registar.
250  output reg mdr_sel);              // Selektroski signal za MDR registar.
```

```
252  reg [1:0] step; // Koraci pri izvrsenju, tj. stanje konacnog automata.
```

Модел контролне јединице

```
254 always @(rst or clk)
255    if(rst==1)
256    begin
257       pc_ena <= `ENABLE;
258       pc_sel <= `PC_SEL_0;
259       step <= `FETCH;
260    end
261    else
```

Процедура која се извршава на сваку промену ресет и такт сигнала. У случају активирања ресет сигнала *PC* регистар се поставља на вредност нула а корак постаља на почетни корак.

```
262    /*Konacni automat (state machine)*/
263    if(clk == 1'b1)
264    case(step)
265       `FETCH:
266          begin
267             a_sel <= `BUS_A_SEL_PC;
268             alu_ena <= `ENABLE;
269             alu_op <= `ALU_INC;
270             mar_ena <= `ENABLE;
271             mdr_ena <= `ENABLE;
272             mdr_sel <= `MDR_SEL_DATA;
273             mem_op <= `MEM_READ;
274             step <= `DECODE;
275          end
```

Од 262 лине кода почиње имплементација коначног аутомата. У првом кораку се врши доношење инструкције из меморије пребацивањем вредности регистра *PC* у *MAR*. У овом кораку се врши и инкрементирање *PC* и на тај начин овај регистра садржи вредност меморијске локације на којој се налази следећа инструкција.

Модел контролне јединице

```
276         `DECODE:
277           begin
278              a_sel <= `BUS_A_SEL_MDR;
279              b_sel <= `BUS_B_SEL_ALU_RES;
280              ir_ena <= `ENABLE;
281              pc_ena <= `ENABLE;
282              pc_sel <= `PC_SEL_BUS_B;
283              step <= `EXECUTE;
284           end
```

У другом кораку текућа инструкција, која је у међувремену донешена из меморије у *MDR* регистар, пребацује у *IR* регистар.

У овом кораку се и инкрементирана вредност *PC* регисата уписује у *PC* регистра.