### CU

module clk\_cu(input [31:0]a,input clk, output iord,output irwr,output pcwr,output pcwrcond,output regdst,output regwr,output alusrca,output[1:0]pcsrc,output [5:0]aluop,

output [1:0]alusrcb,output memtoreg,output memrd,output memwr);

parameter

ADDU = 6'b100001,

SUBU = 6'b100011,

ADD = 6'b100000,

SUB = 6'b100010,

AND = 6'b100100,

OR = 6'b100101,

XOR = 6'b100110,

NOR = 6'b100111,

SLT = 6'b101010,

SLTU = 6'b101011,

SRL = 6'b000010,

SRA = 6'b000011,

SLL = 6'b000000,

SLLV = 6'b000100,

SRLV = 6'b000110,

SRAV = 6'b000111,

JR = 6'b001000,

JALR = 6'b001001,

MULT = 6'b011000,

MULTU = 6'b011001,

DIV = 6'b011010,

DIVU = 6'b011011,

MFHI = 6'b010000,

MFLO = 6'b010010,

MTHI = 6'b010001,

MTLO = 6'b010011,

BREAK = 6'b001101,

SYSCALL = 6'b001100,

TEQ = 6'b110100,

CLZ = 6'b100000,

MUL = 6'b000010,

BLTZ = 5'b00000,

BGEZ = 5'b00001,

ERET = 6'b011000,

MFC0 = 5'b00000,

MTC0 = 5'b00100,

ADDI = 6'b001000,

ADDIU = 6'b001001,

ANDI = 6'b001100,

ORI = 6'b001101,

XORI = 6'b001110,

LW = 6'b100011,

SW = 6'b101011,

BEQ = 6'b000100,

BNE = 6'b000101,

BLEZ = 6'b000110,

BGTZ = 6'b000111,

SLTI = 6'b001010,

SLTIU = 6'b001011,

LUI = 6'b001111,

J = 6'b000010,

JAL = 6'b000011,

LB = 6'b100000,//

LBU = 6'b100100,//

LH = 6'b100001,//

LHU = 6'b100101,//

SB = 6'b101000,//

SH = 6'b101001,//

SPECIAL = 6'b000000,

SPECIAL2= 6'b011100,

REGIMM = 6'b000001,

COP0 = 6'b010000;

//alu

parameter \_ADDU = 4'b0000; //r=a+b unsigned

parameter \_ADD = 4'b0010; //r=a+b signed

parameter \_SUBU = 4'b0001; //r=a-b unsigned

parameter \_SUB = 4'b0011; //r=a-b signed

parameter \_AND = 4'b0100; //r=a&b

parameter \_OR = 4'b0101; //r=a|b

parameter \_XOR = 4'b0110; //r=a^b

parameter \_NOR = 4'b0111; //r=~(a|b)

parameter \_LUI = 4'b1000; //r={b[15:0],16'b0}

parameter \_SLT = 4'b1011; //r=(a-b<0)?1:0 signed

parameter \_SLTU = 4'b1010; //r=(a-b<0)?1:0 unsigned

parameter \_SRA = 4'b1100; //r=b>>>a

parameter \_SLL = 4'b1110; //r=b<<a

parameter \_SRL = 4'b1101; //r=b>>a

parameter \_DIVU = 6'b010000;

parameter \_MULTU = 6'b010001;

parameter \_SYSCALL= 4'b1000,

\_BREAK = 4'b1001,

\_TEQ = 4'b1101;

reg c1,c2,c3,c4,c5,c6,c7,c11,c12,c13;

reg[1:0]c8;

reg[1:0]c10;

reg[3:0]c9;

reg [5:0]code,codesuan;

reg [31:0]i;

reg [31:0]aimcircle;

reg suanshu;

initial

begin

c1=0;c3=1;c2=1;c7=0;c10=1;c9=\_ADDU;c8=0; code=0;c12=1;c13=0;//1th

i=0;

aimcircle=3;

end

always@(posedge clk)

begin

if(i<aimcircle)

begin

case(i)

0:

begin

i=i+1;

c7=0;c10=3;c9=\_ADDU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c12=0;

end

1:

begin

i=i+1;

if (suanshu==0)

case(code)

LW:begin c7=1;c10=2;c9=\_ADDU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SW:begin c7=1;c10=2;c9=\_ADDU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

ADDI:begin c7=1;c10=2;c9=\_ADD; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

ANDI:begin c7=1;c10=2;c9=\_AND; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

ORI:begin c7=1;c10=2;c9=\_OR; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

XORI:begin c7=1;c10=2;c9=\_XOR; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SLTI:begin c7=1;c10=2;c9=\_SLT; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SLTIU:begin c7=1;c10=2;c9=\_SLTU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

ADDIU:begin c7=1;c10=2;c9=\_ADDU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

BEQ:begin c7=1;c10=0;c9=\_ADDU; c1=0;c2=0;c3=0;c4=1;c5=0;c6=0;c8=1;c11=0;c12=0;c13=0;end

J:begin c7=1;c10=0;c9=\_ADDU; c1=0;c2=0;c3=1;c4=0;c5=0;c6=0;c8=2;c11=0;c12=0;c13=0;end

endcase

else

case(codesuan)

ADD:begin c7=1;c10=0;c9=\_ADD; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SUB:begin c7=1;c10=0;c9=\_SUB; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

ADDU:begin c7=1;c10=0;c9=\_ADDU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

AND:begin c7=1;c10=0;c9=\_AND; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

OR:begin c7=1;c10=0;c9=\_OR; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SLT:begin c7=1;c10=0;c9=\_SLT; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

XOR:begin c7=1;c10=0;c9=\_XOR; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

NOR:begin c7=1;c10=0;c9=\_NOR; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SLTU:begin c7=1;c10=0;c9=\_SLTU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SRAV:begin c7=1;c10=0;c9=\_SRA; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SLLV:begin c7=1;c10=0;c9=\_SLL; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SRLV:begin c7=1;c10=0;c9=\_SRL; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

DIVU:begin c7=1;c10=0;c9=\_DIVU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

MULTU:begin c7=1;c10=0;c9=\_MULTU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

SUBU:begin c7=1;c10=0;c9=\_SUBU; c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c8=0;c11=0;c12=0;c13=0;end

endcase

end

2:

begin

i=i+1;

if(suanshu==0)

case(code)

LW:begin c1=1;c2=0;c3=0;c4=0;c5=0;c6=0;c7=0;c8=0;c9=\_ADDU;c10=0;c11=0;c12=1;c13=0;end

SW:begin c1=1;c2=0;c3=0;c4=0;c5=0;c6=0;c7=0;c8=0;c9=\_ADDU;c10=0;c11=0;c12=0;c13=1;end

ADDIU,SLTIU, SLTI,XORI, ORI,ANDI,ADDI:begin c1=0;c2=0;c3=0;c4=0;c5=0;c6=1;c7=0;c8=0;c9=\_ADDU;c10=0;c11=0;c12=0;c13=0;end

endcase

else

case(codesuan)

SUBU,MULTU,DIVU,SRLV,SLLV,SRAV,SLTU,NOR,XOR,SLT,OR,AND,ADDU,SUB,ADD:begin c1=0;c2=0;c3=0;c4=0;c5=1;c6=1;c7=0;c8=0;c9=\_ADDU;c10=0;c11=0;c12=0;c13=0;end

endcase

end

3:

begin

i=i+1;

if (suanshu==0)

case(code)

LW:begin c1=0;c2=0;c3=0;c4=0;c5=0;c6=1;c7=0;c8=0;c9=\_ADDU;c10=0;c11=1;c12=0;c13=0;end

endcase

end

endcase

end

else

begin

c1=0;c2=0;c3=0;c4=0;c5=0;c6=0;c7=0;c8=0;c9=0;c10=0;c11=0;c12=0;c13=0;

c1=0;c3=1;c2=1;c7=0;c10=1;c9=\_ADDU;c8=0; code=0;c12=1;c13=0;//1th

i=0;

end

end

always@(a)

begin

suanshu=0;

code[5:0]=a[31:26];

codesuan[5:0]=a[5:0];

if (code==0)

begin

suanshu=1;

end

case (code)

LW:aimcircle=4;

SW:aimcircle=3;

0:aimcircle=3;

BEQ:aimcircle=2;

J:aimcircle=2;

ADDI,ANDI,ORI,XORI,SLTI,SLTIU,ADDIU:aimcircle=3;

endcase

end

assign iord=c1;

assign irwr=c2;

assign pcwr=c3;

assign pcwrcond=c4;

assign regdst=c5;

assign regwr=c6;

assign alusrca=c7;

assign pcsrc=c8;

assign aluop=c9;

assign alusrcb=c10;

assign memtoreg=c11;

assign memrd=c12;

assign memwr=c13;

endmodule

### 主模块

module testduo;

wire [31:0]s1,s2,c16,s3,s4,c1,s5,s6,c13,c12,s7,s8,c2,s9,s10,s11,c5,s12,s13,s14,s15,s16,c6,s17,s18,s19,s20,s21,c7,c10,s22,c9,c14,s23,s24,s25,c8,c15,c4,c3,c11,pcrstl;

reg clk,int1,pcrst,go;

initial

begin

clk=0;

int1=1;

//pcrst=0;

end

//assign pcrstl=pcrst;

always #50 clk=~clk;

assign s22=int1;

assign c15=c14&c4;

assign c16=c15|c3;

assign s6=s18;

clk\_pc pc(.clk(clk),.wena(c16),.data\_in(s2),.data\_out(s1));

MUX mux1(.a(s1),.b(s3),.crl(c1),.c(s4));

MUXfour mux2(.a(s10),.b(s10),.crl(c5),.c(s12));

MUX mux3(.a(s3),.b(s13),.crl(c11),.c(s14));

MUX mux4(.a(s1),.b(s17),.crl(c7),.c(s21));

MUX4 mux5(.crl(c10),.a(s18),.b(s22),.c(s19),.d(s20),.e(s11));

MUX4 mux6(.crl(c8),.a(s23),.b(s3),.c(s25),.e(s2));

clkram ram1(.wena(c13),.rena(c12),.addr(s4),.data\_in(s6),.data\_out(s5),.clk(clk));

clk\_ir ir(.d(s5),.b(s8),.a(s7),.wena(c2),.clk(clk),.s9(s9),.all(s10));

clk\_temp t1(.clk(clk),.a(s5),.b(s13));

clk\_temp t2(.clk(clk),.a(s16),.b(s17));

clk\_temp t3(.clk(clk),.a(s15),.b(s18));

clk\_temp t4(.clk(clk),.a(s23),.b(s3));

RF rf(.clk(clk),.we(c6),.raddr1(s7),.raddr2(s8),.waddr(s12),.wdata(s14),.rdata1(s16),.rdata2(s15));

sixteentodou se(.a(s10),.b(s19));

//OSHL2 sh1(.a(s19),.b(s20));

//OSHL2 sh2(.a(s9),.b(s24));

assign s24=s9;

assign s20=s19;

alu alu1(.a(s21),.b(s11),.aluc(c9),.zero(c14),.r(s23));

clk\_cu cu(.clk(clk),.a(s10),.iord(c1),.irwr(c2),.pcwr(c3),.pcwrcond(c4),.regdst(c5),.regwr(c6),.alusrca(c7),.pcsrc(c8),.aluop(c9),.alusrcb(c10),.memtoreg(c11),.memrd(c12),.memwr(c13));

assign s25[31:26]=s1[31:26];

assign s25[25:0]=s24[25:0];

endmodule

### 寄存器地址多路选择器

module MUXfour(input [31:0]a,input [31:0]b, input crl,output [4:0]c);

reg [4:0] temp1;

reg [4:0] temp2;

reg [4:0] temp;

always @(\*)

begin

temp1 [4:0] =a[20:16];

temp2 [4:0] =b[15:11];

if(crl==0)

temp=temp1;

else

temp=temp2;

end

assign c=temp;

endmodule

### 二路选择器

module MUX(input [31:0]a,input [31:0]b, input crl,output [31:0]c);

reg [31:0]temp;

always @(\*)

begin

if(crl==0)

temp=a;

else

temp=b;

end

assign c=temp;

endmodule

### 四路选择器

module MUXfour(input [31:0]a,input [31:0]b, input crl,output [4:0]c);

reg [4:0] temp1;

reg [4:0] temp2;

reg [4:0] temp;

always @(\*)

begin

temp1 [4:0] =a[20:16];

temp2 [4:0] =b[15:11];

if(crl==0)

temp=temp1;

else

temp=temp2;

end

assign c=temp;

endmodule

### PC

module clk\_pc(

input clk,

input wena,

input [31:0] data\_in,

output [31:0] data\_out

);

reg [31:0] data;

initial

begin

data=32'h00000000;

end

always @(posedge clk)

if(wena)

data[31:0]<=data\_in[31:0]; //enable ,input

assign data\_out = data;

endmodule

### RAM

module clkram(//IM&&RAM

input wena,

input rena,

input clk,

input [31:0] addr,

input [31:0] data\_in,

output [31:0] data\_out

);

reg [31:0]i;

reg [31:0] state [0:10000];

reg [31:0] whichtoshow;

initial

begin

whichtoshow=32'b0;

end

initial

begin

for(i=0;i<255;i=i+1)

begin

state[i]=32'h00000000;

$readmemb("F:/duo",state);

end

state[200]=999;

end

always @(addr or rena)

begin

if(rena)

whichtoshow=state[addr];

else

whichtoshow=32'bz;

end

always@(posedge clk)

if(wena)

begin

if(addr!=0) state[addr]<=data\_in;

end

assign data\_out=whichtoshow;

endmodule

### IR寄存器

module clk\_ir(input [31:0]d,input clk,input wena,output [4:0]a,output[4:0]b,output[15:0] c ,output [5:0]code,output [25:0]s9,output [31:0]all);

reg [31:0]temps;

always @(posedge clk)

begin

if(wena)

temps=d;

end

initial

temps=32'b0;

assign a[4:0]=temps[25:21];

assign b[4:0]=temps[20:16];

assign c[15:0]=temps[15:0];

assign code[5:0]=temps[31:26];

assign s9[25:0]=temps[25:0];

assign all=temps;

endmodule

### 暂存器

module clk\_temp(input clk,input [31:0]a,output [31:0]b);

reg [31:0] temp;

always@(posedge clk)

begin

temp=a;

end

assign b=temp;

endmodule

### 寄存器组

module RF(

input clk,

input rst,

input we,

input [4:0] raddr1,

input [4:0] raddr2,

input [4:0] waddr,

input [31:0] wdata,

output [31:0] rdata1,

output [31:0] rdata2

);

reg [31:0] data [0:31];

reg [8:0] i;

reg [31:0] a1,a2;

initial

begin

a1=0;

a2=0;

for(i=0;i<32;i=i+1)

begin

begin

data[i]=i;

end

data[3]=-1;

data[4]=2;

data[2]=8;

data[0]=177;

data[5]=177;

data[6]=65793;

end

end

always@(posedge clk or posedge rst)

begin

if(rst) for(i=0;i<32;i=i+1) begin

data[i]<=32'b0;

end

else begin

if(we) data[waddr]<=wdata;

end

end

always@(\*)

begin

a1=data[raddr1];

a2=data[raddr2];

end

assign rdata1=a1;

assign rdata2=a2;

endmodule

### 符号拓展

module sixteentodou(input [31:0]a,output [31:0]b);

reg [31:0]ac;

reg [31:0]i;

initial

begin

ac=0;

i=0;

end

always @(a)

begin

ac[15:0]=a[15:0];

if (a[15]==0)

for (i=16;i<=31;i=i+1)

ac[i]=0;

else

for (i=16;i<=31;i=i+1)

ac[i]=1;

end

assign b=ac;

endmodule