实验1 简单计算机系统基本模块设计A-代码

电 25 吴晨聪 2022010311

# 1.ROM.v

module ROM(clk,rst\_n,q);

input clk;

input rst\_n;

output [15:0]q;

wire [7:0]addr;

addrGen addrGen(

.clk(clk),

.rst\_n(rst\_n),

.addr(addr)

);

cpurom cpu\_rom\_inst(

.address(addr),

.clock(clk),

.q(q)

);

endmodule

# 2. addrGen.v

module addrGen(clk,rst\_n,addr);

input clk;

input rst\_n;

output reg[7:0]addr;

always@(posedge clk)

begin

if (rst\_n == 1)

addr=0;

else begin

addr=addr+1;

if (addr>255) begin

addr=0;

end

end

end

endmodule

# 3. ROM\_tb.v

`timescale 1ns/1ps

module ROM\_tb;

reg clk;

reg rst\_n;

wire [7:0]addr;

wire [15:0]q;

initial begin

clk = 0;

rst\_n = 0;

#10.1

rst\_n = 1;

#50.1

rst\_n = 0;

end

always #10 clk = ~clk;

ROM ROM(

.clk(clk),

.rst\_n(rst\_n),

.q(q)

);

addrGen addrGen(

.clk(clk),

.rst\_n(rst\_n),

.addr(addr)

);

endmodule

# 4. ROM2.v

module ROM2(clk,rst\_n,q);

input clk;

input rst\_n;

output [31:0]q;

wire [7:0]addr;

addrGen addrGen(

.clk(clk),

.rst\_n(rst\_n),

.addr(addr)

);

cpurom2 cpu\_rom\_inst(

.address(addr),

.clock(clk),

.q(q)

);

endmodule

# 5. ROM2\_tb.v

`timescale 1ns/1ps

module ROM2\_tb;

reg clk;

reg rst\_n;

wire [7:0]addr;

wire [31:0]q;

initial begin

clk = 0;

rst\_n = 0;

#10.1

rst\_n = 1;

#50.1

rst\_n = 0;

end

always #10 clk = ~clk;

ROM2 ROM2(

.clk(clk),

.rst\_n(rst\_n),

.q(q)

);

addrGen addrGen(

.clk(clk),

.rst\_n(rst\_n),

.addr(addr)

);

endmodule

# 6. regfile.v

module regfile(clk,rst\_n,n1,n2,nd,di,reg\_we,q1,q2);

input clk;

input rst\_n;

input [1:0]n1;

input [1:0]n2;

input [1:0]nd;

input [7:0]di;

input reg\_we;

output reg[7:0]q1;

output reg[7:0]q2;

reg[7:0] rf[3:0];

initial begin

rf[0] = 10;

rf[1] = 11;

rf[2] = 12;

rf[3] = 13;

end

always@(posedge clk)

begin

if(rst\_n ==1) begin

rf[0] = 0;

rf[1] = 0;

rf[2] = 0;

rf[3] = 0;

end

else begin

q1=rf[n1];

q2=rf[n2];

if(reg\_we == 1) begin

rf[nd]=di;

end

end

end

endmodule

# 7. regfile\_tb.v

`timescale 1ns/1ps

module regfile\_tb;

reg clk;

reg rst\_n;

reg [1:0]n1;

reg [1:0]n2;

reg [1:0]nd;

reg [7:0]di;

reg reg\_we;

wire[7:0]q1;

wire[7:0]q2;

initial begin

clk = 0;

rst\_n = 0;

n1 = 0;

n2 = 0;

nd = 0;

di = 0;

reg\_we = 0;

#100.1 rst\_n = 1;

#50 rst\_n = 0;

#100 nd = 0;

di = 11;

reg\_we = 1;

#100 nd = 0;

di = 12;

#100 nd = 0;

di = 11;

end

always #10 clk = ~clk;

regfile regfile(

.clk(clk),

.rst\_n(rst\_n),

.n1(n1),

.n2(n2),

.nd(nd),

.di(di),

.reg\_we(reg\_we),

.q1(q1),

.q2(q2)

);

endmodule

# 8. regfile2.v

module regfile2(clk,rst\_n,n1,n2,nd,di,reg\_we,q1,q2);

input clk;

input rst\_n;

input [4:0]n1;

input [4:0]n2;

input [4:0]nd;

input [31:0]di;

input reg\_we;

output reg[31:0]q1;

output reg[31:0]q2;

reg[31:0] rf[31:0];

initial begin

rf[0] = 0;

rf[1] = 1;

rf[2] = 2;

rf[3] = 3;

end

always@(posedge clk)

begin

if(rst\_n ==1) begin

rf[0] = 0;

rf[0] = 1;

rf[0] = 2;

rf[0] = 3;

end

else begin

q1=rf[n1];

q2=rf[n2];

if(reg\_we == 1) begin

rf[nd]=di;

end

end

end

endmodule

# 9. regfile2.v

module regfile2(clk,rst\_n,n1,n2,nd,di,reg\_we,q1,q2);

input clk;

input rst\_n;

input [4:0]n1;

input [4:0]n2;

input [4:0]nd;

input [31:0]di;

input reg\_we;

output reg[31:0]q1;

output reg[31:0]q2;

reg[31:0] rf[31:0];

initial begin

rf[0] = 0;

rf[1] = 1;

rf[2] = 2;

rf[3] = 3;

end

always@(posedge clk)

begin

if(rst\_n ==1) begin

rf[0] = 0;

rf[0] = 1;

rf[0] = 2;

rf[0] = 3;

end

else begin

q1=rf[n1];

q2=rf[n2];

if(reg\_we == 1) begin

rf[nd]=di;

end

end

end

endmodule

# 10. regfile2\_tb.v

`timescale 1ns/1ps

module regfile2\_tb;

reg clk;

reg rst\_n;

reg [4:0]n1;

reg [4:0]n2;

reg [4:0]nd;

reg [31:0]di;

reg reg\_we;

wire[31:0]q1;

wire[31:0]q2;

initial begin

clk = 0;

rst\_n = 0;

n1 = 0;

n2 = 0;

nd = 0;

di = 0;a

reg\_we = 0;

#50.1 rst\_n = 1;

#50 rst\_n = 0;

#100 nd = 0;

di = 10;

reg\_we = 1;

#100 nd = 1;

di = 11;

reg\_we = 1;

#100 nd = 2;

di = 12;

reg\_we = 1;

#100 nd = 3;

di = 13;

reg\_we = 1;

#100 n1 = 1;

n2 = 2;

reg\_we = 0;

#100 n1 = 2;

n2 = 3;

end

always #10 clk = ~clk;

regfile2 regfile2(

.clk(clk),

.rst\_n(rst\_n),

.n1(n1),

.n2(n2),

.nd(nd),

.di(di),

.reg\_we(reg\_we),

.q1(q1),

.q2(q2)

);

endmodule

# 11. alu.v

module alu(data\_a,data\_b,s,zero,cs,carry\_in,carry\_out);

input[7:0] data\_a,data\_b;

input[2:0] cs;

input carry\_in;

output reg[7:0]s;

output reg zero;

output reg carry\_out;

always@(\*) begin

case(cs)

3'b000: s = data\_a & data\_b;

3'b001: s = data\_a | data\_b;

3'b010: if(data\_a + data\_b >= 9'b100000000) begin

s = data\_a + data\_b;

carry\_out = 1;

end

else begin

s = data\_a + data\_b;

carry\_out = 0;

end

3'b011: if(data\_a > data\_b) begin

s = data\_a - data\_b;

carry\_out = 1;

end

else begin

s = data\_a - data\_b;

carry\_out = 0;

end

3'b100: if(data\_a < data\_b) s = 1;

else s = 0;

3'b101: if(data\_a > data\_b + 1 - carry\_in) begin

s = data\_a - data\_b - 1 + carry\_in;

carry\_out = 1;

end

else begin

s = data\_a - data\_b - 1 + carry\_in;

carry\_out = 0;

end

3'b110: if(data\_a + data\_b + carry\_in >= 9'b100000000) begin

s = data\_a + data\_b + carry\_in;

carry\_out = 1;

end

else begin

s = data\_a + data\_b + carry\_in;

carry\_out = 0;

end

endcase

if(s == 0) zero = 1;

else zero = 0;

end

endmodule

# 11. alu\_tb.v

`timescale 1ns/1ps

module alu\_tb;

reg[7:0] data\_a,data\_b;

reg[2:0] cs;

reg carry\_in;

wire[7:0]s;

wire zero;

wire carry\_out;

parameter AND =3'b000,

OR =3'b001,

ADD =3'b010,

SUB =3'b011,

SLT =3'b100,

SUBC=3'b101,

ADDC=3'b110;

initial begin

data\_a = 0;

data\_b = 0;

cs = 0;

carry\_in = 0;

#100 data\_a = 8'h1f;

data\_b = 8'hf1;

cs = AND;

#100 cs = OR;

#100 cs = ADD;

#100;

end

alu alu(

.data\_a(data\_a),

.data\_b(data\_b),

.s(s),

.zero(zero),

.cs(cs),

.carry\_in(carry\_in),

.carry\_out(carry\_out)

);

endmodule

# 12. alu2.v

module alu2(data\_a,data\_b,s,zero,cs,carry\_in,carry\_out);

input[31:0] data\_a,data\_b;

input[2:0] cs;

input carry\_in;

output reg[31:0]s;

output reg zero;

output reg carry\_out;

always@(\*) begin

case(cs)

3'b000: s = data\_a & data\_b;

3'b001: s = data\_a | data\_b;

3'b010: if(data\_a + data\_b > 4294967295) begin

s = data\_a + data\_b;

carry\_out = 1;

end

else begin

s = data\_a + data\_b;

carry\_out = 0;

end

3'b011: if(data\_a > data\_b) begin

s = data\_a - data\_b;a

carry\_out = 1;

end

else begin

s = data\_a - data\_b;

carry\_out = 0;

end

3'b100: if(data\_a < data\_b) s = 1;

else s = 0;

3'b101: if(data\_a > data\_b + 1 - carry\_in) begin

s = data\_a - data\_b - 1 + carry\_in;

carry\_out = 1;

end

else begin

s = data\_a - data\_b - 1 + carry\_in;

carry\_out = 0;

end

3'b110: if(data\_a + data\_b + carry\_in > 4294967295) begin

s = data\_a + data\_b + carry\_in;

carry\_out = 1;

end

else begin

s = data\_a + data\_b + carry\_in;

carry\_out = 0;

end

endcase

if(s == 0) zero = 1;

else zero = 0;

end

endmodule

# 13. alu2\_tb.v

`timescale 1ns/1ps

module alu2\_tb;

reg[31:0] data\_a,data\_b;

reg[2:0] cs;

reg carry\_in;

wire[31:0]s;

wire zero;

wire carry\_out;

parameter AND =3'b000,

OR =3'b001,

ADD =3'b010,

SUB =3'b011,

SLT =3'b100,

SUBC=3'b101,

ADDC=3'b110;

initial begin

data\_a = 0;

data\_b = 0;

cs = 0;

carry\_in = 0;

#100 data\_a = 32'h1f;

data\_b = 32'hf1;

cs = AND;

#100 cs = OR;

#100 cs = ADD;

#100 cs = SUB;

#100 cs = SLT;

#100 cs = SUBC;

carry\_in = 0;

#100 cs = ADDC;

carry\_in = 1;

#100;

end

alu2 alu2(

.data\_a(data\_a),

.data\_b(data\_b),

.s(s),

.zero(zero),

.cs(cs),

.carry\_in(carry\_in),

.carry\_out(carry\_out)

);

endmodule