实验2 简单计算机系统基本模块设计B-代码

电 25 吴晨聪 2022010311

# 1. addrDataGen.v

module addrDataGen(clk,rst\_n,wren,aclr,addr,data);

input clk;

input rst\_n;

reg [1:0]count;

output reg wren;

output reg aclr;

output reg[7:0] addr;

output reg[7:0] data;

reg state;

initial begin

aclr=0;

addr=0;

wren=0;

data=1;

count=0;

end

always@(posedge clk or negedge rst\_n)

begin

if(!rst\_n) begin

aclr=1;

addr=0;

wren=0;

data=1;

count=0;

end

else begin

count=count+1;

if(count>=3)begin

count=0;

aclr=0;

addr=addr+1;

data=data+1;

if(addr>10) begin

addr=0;

end

if(data>15) begin

data=0;

end

wren=1-wren;

end

end

end

endmodule

# 2. RAM.v

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

input clk;

input rst\_n;

output [7:0]q;

wire aclr;

wire wren;

wire [7:0]data;

wire [7:0]addr;

addrDataGen addrDataGen(

.clk(clk),

.rst\_n(rst\_n),

.aclr(aclr),

.wren(wren),

.addr(addr),

.data(data)

);

cpuram cpu\_ram\_inst(

.aclr(aclr),

.wren(wren),

.address(addr),

.data(data),

.clock(clk),

.q(q)

);

endmodule

# 3. RAM\_tb.v

`timescale 1ns/1ps

module RAM\_tb;

reg clk;

reg rst\_n;

wire aclr;

wire [7:0]addr;

wire [7:0]data;

wire wren;

wire [7:0]q;

initial begin

clk = 0;

rst\_n = 1;

#10.1

rst\_n = 0;

#50.1

rst\_n = 1;

end

always #10 clk = ~clk;

RAM RAM(

.rst\_n(rst\_n),

.clk(clk),

.q(q)

);

addrDataGen addrDataGen(

.clk(clk),

.rst\_n(rst\_n),

.aclr(aclr),

.wren(wren),

.addr(addr),

.data(data)

);

endmodule

# 4. addrDataGen2.v

module addrDataGen2(clk,rst\_n,wren,aclr,addr,data);

input clk;

input rst\_n;

reg [1:0]count;

output reg wren;

output reg aclr;

output reg[7:0] addr;

output reg[31:0] data;

reg state;

initial begin

aclr=0;

addr=0;

wren=0;

data=1;

count=0;

end

always@(posedge clk or negedge rst\_n)

begin

if(!rst\_n) begin

aclr=1;

addr=0;

wren=0;

data=1;

count=0;

end

else begin

count=count+1;

if(count>=3)begin

count=0;

aclr=0;

addr=addr+1;

data=data+1;

if(addr>10) begin

addr=0;

end

if(data>15) begin

data=0;

end

wren=1-wren;

end

end

end

endmodule

# 5. RAM2.v

module RAM2(clk,rst\_n,q);

input clk;

input rst\_n;

output [31:0]q;

wire aclr;

wire wren;

wire [31:0]data;

wire [7:0]addr;

addrDataGen2 addrDataGen2(

.clk(clk),

.rst\_n(rst\_n),

.aclr(aclr),

.wren(wren),

.addr(addr),

.data(data)

);

cpuram2 cpu\_ram\_inst2(

.aclr(aclr),

.wren(wren),

.address(addr),

.data(data),

.clock(clk),

.q(q)

);

endmodule

# 6. RAM2\_tb.v

`timescale 1ns/1ps

module RAM2\_tb;

reg clk;

reg rst\_n;

wire aclr;

wire [7:0]addr;

wire [31:0]data;

wire wren;

wire [31:0]q;

initial begin

clk = 0;

rst\_n = 1;

#10.1

rst\_n = 0;

#50.1

rst\_n = 1;

end

always #10 clk = ~clk;

RAM2 RAM2(

.rst\_n(rst\_n),

.clk(clk),

.q(q)

);

addrDataGen2 addrDataGen2(

.clk(clk),

.rst\_n(rst\_n),

.aclr(aclr),

.wren(wren),

.addr(addr),

.data(data)

);

endmodule

# 7. flag.v

module flag(clk,rst\_n,flagwrite,flagin,flagout,zeroin,zeroout);

input clk;

input rst\_n;

input flagwrite;

input flagin;

input zeroin;

output reg flagout;

output reg zeroout;

always@(posedge clk or negedge rst\_n) begin

if(! rst\_n) begin

flagout=0;

end

else begin

if(flagwrite) begin

flagout=flagin;

end

zeroout=zeroin;

end

end

endmodule

# 8. flag\_tb.v

`timescale 1ns/1ps

module flag\_tb;

reg clk;

reg rst\_n;

reg flagwrite;

reg flagin;

wire flagout;

initial begin

clk = 0;

rst\_n = 0;

flagwrite = 0;

flagin = 0;

#50.1 rst\_n = 1;

#100 flagin = 1;

#100 flagwrite = 1;

#100 flagwrite = 0;

flagin = 0;

end

always #10 clk = ~clk;

flag flag(

.clk(clk),

.rst\_n(rst\_n),

.flagwrite(flagwrite),

.flagin(flagin),

.flagout(flagout)

);

endmodule

# 9. cpuA.v

module cpuA(scrA,scrB,alucs,clk,rst\_n,flagwrite,s,zeroout);

input [7:0]scrA;

input [7:0]scrB;

input [2:0]alucs;

input clk;

input rst\_n;

input flagwrite;

wire carry\_out;

wire carry\_in;

wire zeroin;

output [7:0]s;

output zeroout;

always@(posedge clk or negedge rst\_n) begin

if(!rst\_n) begin

end

else begin

end

end

alu alu(

.data\_a(scrA),

.data\_b(scrB),

.s(s),

.zero(zeroin),

.cs(alucs),

.carry\_in(carry\_in),

.carry\_out(carry\_out)

);

flag flag(

.clk(clk),

.rst\_n(rst\_n),

.zeroin(zeroin),

.flagwrite(flagwrite),

.flagin(carry\_out),

.flagout(carry\_in),

.zeroout(zeroout)

);

endmodule

# 10. cpuA\_tb.v

`timescale 1ns/1ps

module cpuA\_tb;

reg[7:0] scrA,scrB;

reg[2:0] alucs;

reg clk;

reg rst\_n;

reg flagwrite;

wire[7:0]s;

wire zeroout;

parameter AND =3'b000,

OR =3'b001,

ADD =3'b010,

SUB =3'b011,

SLT =3'b100,

SUBC=3'b101,

ADDC=3'b110;

initial begin

clk = 1;

scrA = 0;

scrB = 0;

alucs = 0;

flagwrite = 0;

rst\_n = 0;

#20 rst\_n=1;

#20 alucs=2;

flagwrite=1;

scrA=22;

scrB=11;

#20 scrB=250;

#20 alucs=6;

scrB=33;

#20 alucs=3;

scrB=11;

#20 scrB=253;

#20 alucs=5;

scrB=11;

#20 scrB=10;

#20 alucs=0;

flagwrite=0;

scrA=6;

scrB=5;

#20 alucs=1;

#20;

end

always #10 clk = ~clk;

cpuA cpuA(

.scrA(scrA),

.scrB(scrB),

.alucs(alucs),

.clk(clk),

.rst\_n(rst\_n),

.flagwrite(flagwrite),

.s(s),

.zeroout(zeroout)

);

endmodule