



DIGITAL SYSTEM DESIGN

Professor :

BAYAT

آزمایش 8

Hooman keshvari

99105667

ALIREZA FOROODNIA

99105645



مقدمه :

در آزمایش یک واحد ALU برای اعداد مختلط طراحی شده است که دارای پایپلاین 4 مرحله ای (شامل خواندن دستور، لود کردن operand ها، انجام عملیات های مربوط به ضرب و جمع اعداد مختلط، store کردن خروجی واحد arithmetic در مموری) است.

جزئیات پیاده سازی :

ماژول mul :

module MUL(

 input clk,rst,

 input [7:0] in1,in2,

 output reg [7:0] out,

 output reg is\_done

);

reg signed [3:0] a,b,c,d;

reg [1:0] counter;

always @(posedge clk,negedge rst) begin

    if(rst == 0) begin

      counter = 0;

    end

    else begin

    if(counter < 5) begin

        counter = counter + 1;

        is\_done = 0;

    end

    if(counter == 5) begin

        a = in1[7:4];

        b = in1[3:0];

        c = in2[7:4];

        d = in2[3:0];

        out[3:0] = a\*d + b\*c;

        out[7:4] = a\*c - b\*d;

        is\_done = 1;

        counter = 0;

     end

    end

end

endmodule

این ماژول به صورت اسنکرون ریست می پذیرد. سپس هر ورودی ای که به آن وارد شود طی پنج کلاک عملیات ضرب اعداد مختلط را بر روی ان صورت داده و سپس نتایج ضرب را به همراه سیگنال is\_done به خروجی می دهد.

ماژول add\_sub :

module ADD\_SUB(

 input clk,rst,

 input [7:0] in1,in2,

 input [1:0] add\_or\_sub,

 output reg [7:0] out,

 output reg is\_done

);

reg signed [3:0] a,b,c,d;

reg counter;

always @(posedge clk,negedge rst) begin

  if(rst == 0) begin

    counter = 0;

    is\_done = 0;

  end

  else begin

    if(counter < 2) begin

    counter = counter + 1;

    is\_done = 0;

    end

    if(counter == 2) begin

    a = in1[7:4];

    b = in1[3:0];

    c = in2[7:4];

    d = in2[3:0];

    if(add\_or\_sub == 1) begin

      out[7:4] = a - c;

      out[3:0] = b - d;

    end else if(add\_or\_sub == 0) begin

      out[7:4] = a + c;

      out[3:0] = b + d;

    end

    is\_done = 1;

    counter = 0;

  end

  end

end

endmodule

این ماژول به کمک ورودی add\_or\_sub ابتدا تشخیص می دهد که کدام یک از اعمال جمع یا تفریق باید صورت بپذیرد و سپس طی دو کلاک نتیجه اعمال تشخیص داده شده را به خروجی ها منتقل کرده و سیگنال is\_done را یک می کند.

ماژول های data\_mem و instruction\_mem :

این ماژول ها به ترتیب مسئولیت حفظ داده های عملیات ها و دستورات را بر عهده دارند. شیوه طراحی آنها به این گونه است که دارای ریست آسنکرون می باشند و در صورت رخ دادن ریست به جای صفر شدن داده ها مقادیری را که برای انجام عملیات ها لازم دارند به درون آنها منتقل می کنیم.

module inst\_mem(

    input clk,

    input rst,

    input enable,

    input read\_writenot,

    input [19 : 0] in\_data,

    input [4 : 0] read\_address,

    input [4 : 0] write\_address,

    output reg [19 : 0] out\_data

);

reg [19 : 0] storage [31 : 0];

integer i;

always @(negedge rst) begin

    for (i = 0; i<20 ; i = i + 1 ) begin

        storage[i] = 0;

    end

    storage[0] = 20'b 00\_000000\_000001\_000010;

    storage[1] = 20'b 00\_000011\_000100\_000101;

    storage[2] = 20'b 00\_000110\_000111\_001000;

    storage[3] = 20'b 01\_001001\_001010\_001011;

    storage[4] = 20'b 01\_001100\_001101\_001110;

    storage[5] = 20'b 10\_001111\_010000\_010001;

    storage[6] = 20'b 10\_010010\_010011\_010100;

    storage[7] = 20'b 11\_000000\_000000\_000000;

end

always @(posedge clk) begin

    if(enable) begin

        if(read\_writenot) begin

            out\_data = storage[read\_address];

        end

        else begin

            storage[write\_address] = in\_data;

        end

    end

end

endmodule

ماژول های LD, IF, EX :

این ماژول ها از تعدادی بافر که خروجی ها را به ورودی ها منتقل می کنند تشکیل شده اند تا نقش انتقال داده ها را در پایپلاین اجرا کنند. همه آنها از یک سیگنال freeze مشترک نیز استفاده می کنند تا در مواقع لازم (فاصله زمانی اجرای یک دستور ضرب یا جمع یا تفریق که باید سیستم متوقف باشد) بتوان از عملکرد آن جلوگیری کرد.

module EX(

    // inputs

    input ld\_inst\_halt,

    input halted,

    input [7:0] alu\_output,

    input [5:0] write\_addr,

    input freeze,

    input clk,

    input data\_rw,

    input data\_mem\_write\_ex,

    // outputs

    output reg halted\_out,

    output reg data\_rw\_out,

    output reg [7:0] alu\_output\_out,

    output reg [5:0] write\_addr\_out,

    output reg data\_mem\_write\_out\_ex

);

always @(posedge clk) begin

  if(!freeze && !ld\_inst\_halt) begin

    halted\_out = halted;

    data\_rw\_out = data\_rw;

    alu\_output\_out = alu\_output;

    write\_addr\_out = write\_addr;

    data\_mem\_write\_out\_ex = data\_mem\_write\_ex;

  end

end

endmodule

ماژول data\_path :

این ماژول شامل اتصالات ما بین تمام اجزائ سیستم طراحی شده از جمله پایپلاین ها، واحد کنترل، واحد جمع و تفریق و ضرب و واحد های مموری داده و دستور است.

نحوه کار کلی سیستم :

با هر کلاک ابتدا دستور از حافظه مرتبط با دستور ها خوانده شده و سپس به اولین واحد پایپلاین منتقل می شود.

سپس در ادامه operand ها از داده ها جدا شده و برای گرفته شدن از حافظه داده به آن وارد می شوند. همچنین opcode دستور وارد واحد کنترل و پایپلاین می شود تا در ادامه به alu نیز منتقل بشود.

در قسمت مربوط به alu پایپلاین operand های گرفته شده از مموری در مرحله قبل را به علاوه opcode گرفته شده را به ALU وروی می دهد. واحد arithmetic نیز پس از انجام عملیات مربوطه با یک کردن سیگنال is\_done به بقیه مدار اجازه فعالیت می دهد و خروجی های محاسبات را راهی پایپلاین می کند.

در نهایت پایپلاین آخر خروجی محاسبات را به علاوه آدرس ذخیره که از مرحله دوم تا اینجا پایپ شده است را به واحد حافظه می دهد تا این واحد آنها را ذخیره کرده و یک عملیات خاتمه بیابد.

فعالیت مدار همواره تحت تاثیر سیگنال is\_done خروجی از ALU می باشید زیرا تا زمانی که این واحد نتیجه را آماده نکرده باشد نباید پایپلاین به فعالیت دادمه دهد.

واحد کنترل بسته به نوع opcode وظیفه متوقف کردن مدار یا ایجاد سیگنال هایی که برای انجام آن فرمان ضروری هستند را دارد. این سیگنال ها در صورت لزوم در طول مدار پایپ می شود تا به مقصد خود برسند.

تست بنچ :

module tb\_;

reg clk;

reg rst;

wire halted;

CPU uut(

    .rst (rst),

    .clk (clk),

    .halted(halted)

);

localparam CLK\_PERIOD = 2;

always #(CLK\_PERIOD/2) clk=~clk;

initial begin

    rst = 0;

    clk = 1;

    #1

    rst = 1;

    @(posedge halted)

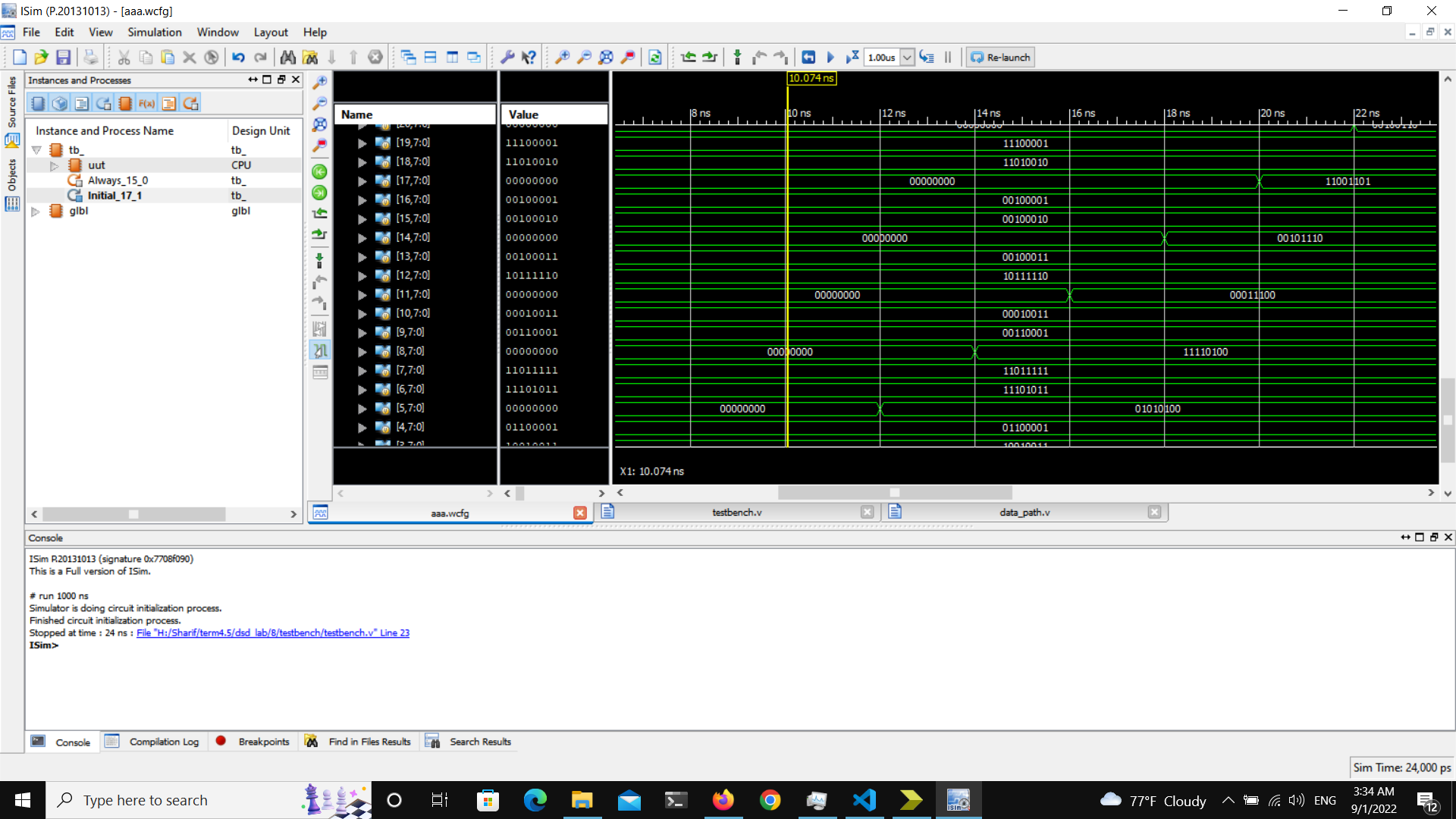
    $finish();

end

endmodule

در تست بنچ ابتدا با ریست شدن مدار مقادیر به واحد های حافظه منتقل می شوند و سپس با ضربان های کلاک مدار فعالیت کرده و خروجی ها را ایجاد می کند.

شکل موج :



گزارش سنتز :

