《计算机组成原理》实验报告

年级、专业、班级	2021 级计算机科学与技术 04、05 班	**姓名**	胡鑫,冯宇馨
实验题目	实验一简单流水线与运算器实验		
实验时间	2023 年 03 月 27 日	**实验地点**	DS1410
实验成绩	优秀/良好/中等	**实验性质**	□验证性 ☑设计性 □综合性

教师评价：

☑算法/实验过程正确； ☑源程序/实验内容提交； ☑程序结构/实验步骤合理;

☑实验结果正确; ☑语法、语义正确; ☑报告规范;

其他:

评价教师: 冯永

实验目的

(1)理解流水线 (Pipeline) 设计原理;

(2)了解算术逻辑单元 ALU 的原理;

(3)熟悉并运用 Verilog 语言设计 ALU;

(4)熟悉并运用 Verilog 语言设计流水线全加器;

报告完成时间: 2023 年 4 月 23 日

1 实验内容

1.1 ALU 设计实验

实验要求实现以下算术运算功能，其对应的控制码及功能如下：

$F_{2:0}$	功能	$F_{2:0}$	功能
000	A + B(Unsigned)	100	$\overline{A}$
001	A - B	***101***	***SLT***
010	A AND B	110	未使用
011	A OR B	111	未使用

表 1: 算数运算控制码及功能

实验要求：

1. 根据 ALU 原理图，使用 Verilog 语言定义 ALU 模块，其中输入输出端口参考实验原理，运算指令码长度为 [2:0]。
2. 仿真时 B 端口输入为 32h'01，A 端口输入参照 4.1 中表格
3. 实现 SLT 功能。
4. 验证表 1中所有功能。
5. 给出 RTL 源程序（.v 文件）

1.2 流水线实验

本次实验为仿真实验，设计完成后仅需进行**行为仿真**。

实验要求：

1. 实现 4 级流水线 8bit 全加器，需带有流水线暂停和刷新；
2. 模拟流水线暂停，仿真时控制 10 周期后暂停流水线 2 周期（第 2 级），流水线恢复流动；
3. 模拟流水线刷新，仿真时控制 15 周期时流水线刷新（第 3 级）。

2 实验设计

2.1 ALU

2.1.1 功能描述

算术逻辑单元接收不同功能的控制信号，执行对应的加法、减法、与、或、非、小于则置位功能后输出结果。

2.1.2 接口定义

信号名	方向	位宽	功能描述
num1	input	8	第一个数据输入
num2	input	32	第二个数据输入
op	input	3	ALU 算术运算操作码
res	output	32	数据输出

2.1.3 逻辑控制

op=000 时, 实现 res=A+B; op=001 时, 实现 res=A-B; op=010 时, 实现 res=A&B; op=011 时, 实现 res=A|B; op=100 时, 实现非 A; op=101 时, 若 A<B, 则将 res 置为 1, 否则将 res 置为 0.

2.2 有阻塞 4 级 8bit 全加器

2.2.1 功能描述

将组合逻辑系统地分割, 并在各个部分（分级）之间插入寄存器, 并暂存中间数据, 目的是将一个大操作分解成若干的小操作, 每一步小操作的时间较小, 从而提高频率, 并且每个小操作可以并行执行, 就能提高数据吞吐率。

2.2.2 接口定义

信号名	方向	位宽	功能描述
cin_a	input	32	第一个数据输入
cin_b	input	32	第二个数据输入
c_in	input	1	进位标志
rst	input	1	刷新信号
clk	input	1	时钟信号
stop	input	1	暂停信号
c_out	output	1	溢出标志
sum	output	32	输出的和

2.2.3 逻辑控制

用 pipe(X)-valid 表示第 X 级流水级上当前存有有效的数据， cout(X) 代表每一级的进位, sum(X) 代表着每一级的结果, pipe(X)-allowin 表示第 X 级能否接受上一级的数据, pipe(X)-ready-go 表示第 X 级能否传递给下一级, pipe(X)-to-pipe(X+1)-valid 表示 pipe(X) 能否进入下一级。当 pipe(X)-to-pipe(X+1)-valid 有效并且 pipe(X+1)-allowin 有效的时候, 可以传给下一级。

3 实验过程记录

记录实验的过程,完成了什么样的工作,存在的问题包括哪些,解决方案如何等。subsubsection 名称自行设定。

3.1 问题 1: 生成比特流失败

问题描述:在选择 Project device 的时候选择了不合适的硬件设备,导致了选择 I/O 端口时,与 constr.xdc 文件无法对应,无法生成比特流

解决方案:对照板子型号在 tool 中进行更正

3.2 问题 2: 综合失败

问题描述:top 文件没有给 display 模块指定实例名,导致综合失败

解决方案:在 top 文件中给 display 模块写上实例名

4 实验结果及分析

4.1 ALU 验证实验结果

操作	Num1	Result
A + B(Unsigned)	8'b00000010	8'b00000011
A - B	8'b11111111	8'b11111110
A AND B	8'b11111110	8'b00000000
A OR B	8'b10101010	8'b10101011
$\overline{A}$	8'b11110000	8'b00001111
SLT	8'b10000001	8'b00000000

表 2: ALU 结果表

4.2 流水线阻塞(暂停)仿真图

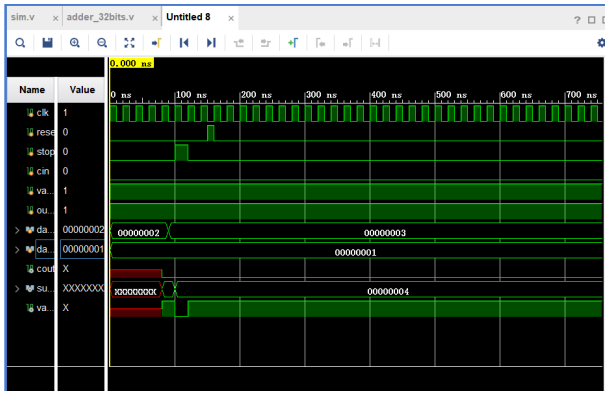

图 1: 流水线阻塞(暂停)仿真图

4.3 流水线刷新(清空)仿真图

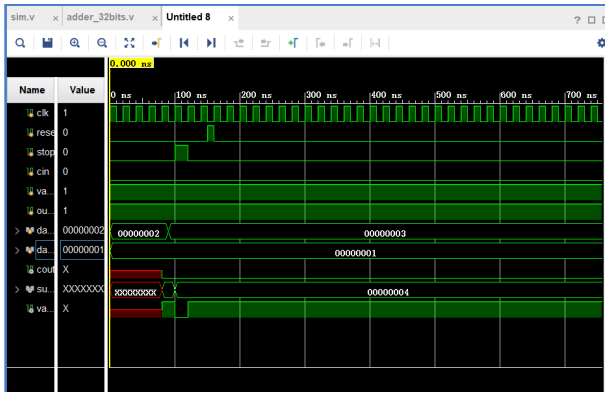

图 2: 流水线(刷新)仿真图

A ALU 代码

```
module ALU(
  input wire [7:0] num1,
  input wire [31:0]num2,
  input wire [2:0] op,
  output reg [31:0] res
    );
    reg [31:0] sign_extend;
    always@ (op)
    begin
    sign_extend={24'h000000,num1[7:0]};
      case(op)
          3'b000: res=sign_extend+num2;
          3'b001: res=sign_extend-num2;
          3'b010: res=sign_extend&num2;
```

```
            3'b011: res=sign_extend|num2;
            3'b100: res=~sign_extend;
            3'b101:
            begin
            if(sign_extend<num2)
                res=1;
            else res=0;
            end
            default:res=32'h00000000;
        endcase
    end
endmodule
```

B 32bit 流水线全加器代码

```
module adder_32bits(
  input wire clk,
  input wire reset,//刷新
  input wire stop,//暂停
  input wire cin,
  input wire valid_in,//判断输入是否有效
  input wire out_allow,//判断是否能输出
  input wire [31:0] data1,
  input wire [31:0] data2,
  output wire cout,//溢出
  output wire [31:0]sum,
  output wire valid_out//判断输出是否有效

    );

//pipeX_valid,表示第 X 级流水级上当前存有有效的数据
  reg pipe1_valid;
  reg pipe2_valid;
  reg pipe3_valid;
  reg pipe4_valid;
  //每一级的进位
  reg cout1;
  reg cout2;
  reg cout3;
  reg cout4;
   //每一级的结果
  reg [7:0]sum1;
  reg [15:0]sum2;
  reg [23:0]sum3;
  reg [31:0]sum4;
  //pipeX_allowin:第X级能否接受上一级的数据
  wire pipe1_allowin;
```

```
 wire pipe2_allowin;
 wire pipe3_allowin;
 wire pipe4_allowin;
 //pipeX_ready_go:第X级能否传给下一级
 wire pipe1_ready_go;
 wire pipe2_ready_go;
 wire pipe3_ready_go;
 wire pipe4_ready_go;
 //pipeX_to_pipex+1_valid:pipeX能否进入下一级
 wire pipe1_to_pipe2_valid;
 wire pipe2_to_pipe3_valid;
 wire pipe3_to_pipe4_valid;

//1
 assign pipe1_ready_go=!stop;//暂停则不能传给下一级
 //pipe1的值无效，或要传给下一级（防止数据丢失），那么可以进行接收
 assign pipe1_allowin=!pipe1_valid || (pipe1_ready_go && pipe2_allowin);
 //pipe1有效且能传递给下一级，所以能进入下一级
 assign pipe1_to_pipe2_valid=pipe1_valid && pipe1_ready_go;

 always @(posedge clk)
 begin
 if(reset)
    pipe1_valid<=1'b0;
 else if (pipe1_allowin) //可以接受
     pipe1_valid<=valid_in;
 //输入有效且可以接受
 if( valid_in&&pipe1_allowin)
 {cout1,sum1}={1'b0,data1[7:0]}+{1'b0,data2[7:0]}+cin;
 end



 //2
assign pipe2_ready_go=!stop;
assign pipe2_allowin=!pipe2_valid || (pipe2_ready_go && pipe3_allowin);
assign pipe2_to_pipe3_valid=pipe2_valid && pipe2_ready_go;

always @(posedge clk)
 begin
 if(reset)
    pipe2_valid<=1'b0;
 else if (pipe2_allowin) //可以接受
     pipe2_valid<=pipe1_to_pipe2_valid;
 //输入有效且可以接受
 if( pipe1_to_pipe2_valid&&pipe2_allowin)
 {cout2,sum2}={{1'b0,data1[15:8]}+{1'b0,data2[15:8]}+cout1,sum1};
 end
```

```
  //3
 assign pipe3_ready_go=!stop;
 assign pipe3_allowin=!pipe3_valid || (pipe3_ready_go && pipe4_allowin);
 assign pipe3_to_pipe4_valid=pipe3_valid && pipe3_ready_go;

 always @(posedge clk)
  begin
  if(reset)
     pipe3_valid<=1'b0;
  else if (pipe3_allowin) //可以接受
      pipe3_valid<=pipe2_to_pipe3_valid;
  //输入有效且可以接受
  if( pipe2_to_pipe3_valid&&pipe3_allowin)
  {cout3,sum3}={{1'b0,data1[23:16]}+{1'b0,data2[23:16]}+cout2,sum2};
  end

 //4
 assign pipe4_ready_go=!stop;
 assign pipe4_allowin=!pipe4_valid || (pipe4_ready_go && out_allow);

 always @(posedge clk)
  begin
  if(reset)
     pipe4_valid<=1'b0;
  else if (pipe4_allowin) //可以接受
      pipe4_valid<=pipe3_to_pipe4_valid;
  //输入有效且可以接受
  if( pipe3_to_pipe4_valid&&pipe4_allowin)
  {cout4,sum4}={{1'b0,data1[31:24]}+{1'b0,data2[31:24]}+cout3,sum3};
  end

  assign cout=cout4;
  assign sum=sum4;
  assign valid_out=pipe4_valid&&pipe4_ready_go;
endmodule
```