Verilog 全加（减）器及累加器设计

TZHUANG 2021-03-04 FPGA 0 COMMENTS

在CPU的ALU的设计中，全加器是ALU的重要组件，因此这部分内容会单独拿出来讲解。

在一位全加器及超前进位加法器的设计技巧中已经将底层算法和逻辑设计步骤介绍清楚，这里主要介绍在FPGA的设计中，利用QuartusII 及Vivado工具进行可综合的设计。

1. **全加器设计**

例1：

```
module  fadd
#(parameter WIDTH = 32)
(
    input              ci,
    input  [WIDTH-1:0] a,
    input  [WIDTH-1:0] b,

    output [WIDTH-1:0] sum,
    output             co
);

assign  {co, sum} = a + b + ci;

endmodule
```

2.**全减器设计**：
全减器采用补码的方式实现，2进制补码对变量取反加一。

其他操作

管理站点

注销

Search ...

本分类下文章

7 Series GTP Transceiver

SPI EEPROM AT25040B

标准SPI 协议

SPI 通讯协议（3）SPI FLASH 介绍

AD9467芯片datasheet

RISC-V LSU，SRAM，GPIO模块（1）exu_lsu模块

数据选择器Verilog的不同设计方法

FPGA中 同步复位（sync）和异步复位（async）的使用

取补码后，原来的减法就变成加法。如： a – b – ci,分别对b和ci取补码，就变成a + com_b + com_ci;

其中b的补码：com_b = ~b+1;

ci的补码：com_ci = ~{(WIDTH-1){1'b0}, ci} + 1;
如果ci==0;　com_ci = {WIDTH-1{1'b1}, 1'b1} + 1 = {WIDTH{1'b1}} + 1 = {WIDTH{1'b0}} = {WIDTH{ci}};
如果ci==1'b1; com_ci = {WIDTH-1{1'b1}, 1'b0} + 1 = {WIDTH{1'b1}} = {WIDTH{ci}};
因此ci的补码统一运算公式就是直接将ci位扩展即可。
例2：

```
module sub
#( parameter WIDTH = 32)
(
    input               ci,
    input [WIDTH-1:0]   a,
    input [WIDTH-1:0]   b,

    output [WIDTH-1:0] sub,
    output              co
);

wire [WIDTH-1:0] com_b;

assign    com_b = ~b + 1;

assign {co, sub} = a + com_b + {WIDTH {ci}};

endmodule
```

3.**累加器的设计**

累加器应用非常广泛，特别是在数学运算，数据平滑，平均值，中值等领域有着广泛应用。

下面以参数的模块为例介绍累加器设计。

例2：

Nand FLASH ECC 算法资料

RISC-V 硬件设计常见问题及解答（2）

Verilog 语法练习（10）复杂计数器的设计

SPI 通讯协议（7）SPI FLASH (verilog) 工程解析 (TOP.v)及仿真

SPI 通讯协议（1）SPI定义与应用

PRA006/010 介绍

异步串行收发（UART）协议详解（1）

Xilinx 7系列FPGA配置官方英文文档

FPGA及硬件设计中的竞争与冒险

FPGA DDR3控制器

Verilog 计数器设计及触发器模型深度分析

网络物理层基本知识（PCS子层）

Vivado Design Suite User Guide — Synthesis

xilinx 公司 7系列 MultiBoot 启动SPI Flash

Verilog语言编程与FPGA应用教材

累加器设计，要求参数化，将输入数据在start信号为'1'后累加，输入数据与data_rdy同步，其中data_rdy是输入数据的同步脉冲。输入数据宽度，以及每次累加的输入数据数量都由参数确定。参数可以接收例化模块传入。累加结束后由outdata输出，并且跟随指示数据计算完成的同步脉冲。

在程序设计中首先将输入数据，输入同步信号锁存，主要用于提高时序。整个程序分为三步，

（1）初始化并等待start信号

（2）累加并将数据累加计数器减1，直到num=0;

（3）累加完成，并回握手信号；返回步骤（1）

程序流程图如下：

变量类型及使用范围深入探讨

Intel® Cyclone® 10 LP 器件数据手册

网络物理层基础知识2（PMA/PMD）

Verilog中的时间尺度与延迟

提高数字系统设计性能–数字设计与Verilog 设计技巧一算法

Verilog 变量声明与数据类型三（可变的向量域选择）

FII-PRA006/010 固化/烧录程序到FLASH

近期文章

Sytem Verilog 教学(视频)

System Verilog 教材

系统验证基础–OVM与UVM

第六节 三极管的特性曲线及参数

第五节 三极管电流分配关系及电流放大系数

第四节 半导体三极

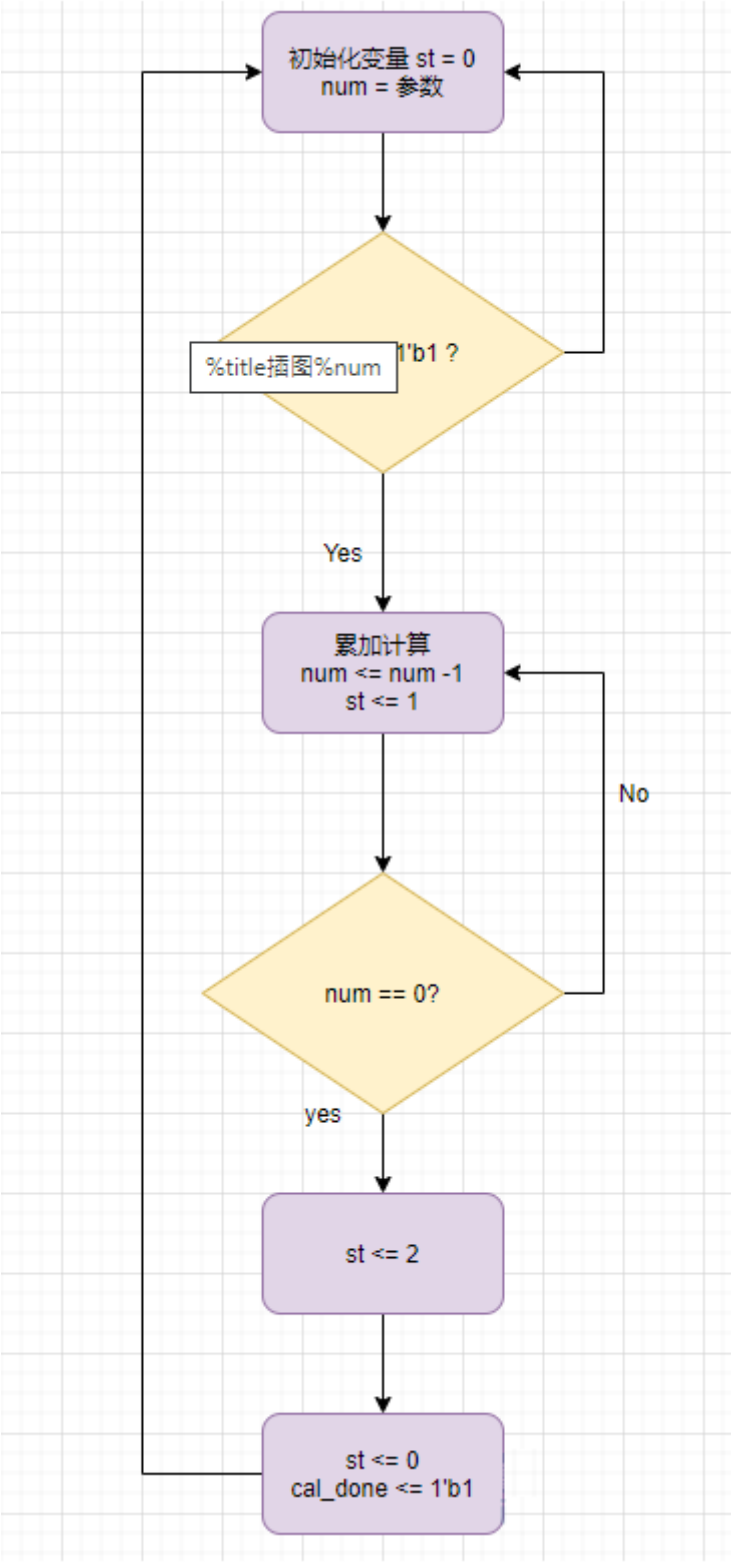

图1

```
1.  module acc #
2.  (
3.      parameter width = 32,
4.      parameter len   = 16
```


分类

选择分类

本周热门文章

```
)
(
    input                                rst,
    input                                clk,
    input                                start,
    input
data_rdy,
    input       [width-1:0]              indata,
    output reg [width+$clog2(len)-1:0]
outdata,
    output reg                           cal_done
);


reg [width-1:0] indata_r;
reg [$clog2(len)-1:0] num;
reg [1:0] st;
reg data_rdy_r;

always@(posedge clk or posedge rst)
if(rst) begin
    indata_r <= 0;
    data_rdy_r <= 0;
end
else begin
    data_rdy_r <= data_rdy;
    indata_r <= indata;
end


always@(posedge clk or posedge rst)
if(rst) begin
     outdata <= 0;
     num <= len;
     cal_done <= 1'b0;
     st <= 0;
end
else case(st)
0:begin
    outdata <= 0;
    num <= len;
    cal_done <= 1'b0;

    if(start)
        st <=1;
end
```

Verilog 中not ~ ！的区别

电流镜 (Current Mirror)

RISC-V指令集讲解（2）I-Type整数寄存器-立即数指令

System Verilog 教材

Verilog 文件操作-$fgetc，$fgets，$fscanf，$fread

互联网网线及接口的基本知识

Verilog仿真中fork...join的原理和使用

```
1:begin
    if(data_rdy_r)begin
        outdata <= outdata + indata_r;
        num <= num - 1;
    end

    if(num == 0) begin
        st <= 2;
    end
end
2:begin
    cal_done <= 1'b1;
    st <= 0;
end
default: st = 0;
endcase

endmodule
```

仿真程序：仿真程序提供自然数1，2，3... n, 每10个数为一批数据，观察计算结果，并提供仿真程序与实体程序之间的各个握手信号。

```
`timescale 1ns/1ps

module tb();


parameter width = 8;
parameter len   = 10;

reg         rst;
reg         clk;
reg         start;
reg         data_rdy;
reg   [width -1:0] indata;
reg   [$clog2(len)-1:0] num;
wire  [width+$clog2(len)-1:0] outdata;
wire  cal_done;



always #5 clk=!clk;


```

```
23.  reg     [2:0] st;
24.
25.
26.  initial  begin
27.      rst = 1'b1;
28.      clk = 1'b0;
29.      #20;
30.      rst = 0;
31.  end
32.
33.  always@(posedge clk or posedge rst)
34.  if(rst) begin
35.      start <= 0;
36.      data_rdy <= 0;
37.      indata <= 0;
38.      num <= 0;
39.      st <= 0;
40.  end
41.  else begin
42.      case (st)
43.      0:begin
44.          data_rdy <= 0;
45.          indata <= 0;
46.          num <= 0;
47.          start <= 1'b1;
48.          st <= 1;
49.      end
50.      1:begin
51.          start <= 0;
52.          if(num == 10) begin
53.              data_rdy <= 0;
54.              num <= 0;
55.              st <= 2;
56.          end
57.          else begin
58.              data_rdy <= 1'b1;
59.              num <= num + 1;
60.              indata <= indata + 1;
61.          end
62.
63.      end
64.      2:begin
65.          st <= 3;
66.      end
67.      3:begin
68.          st <= 4;
```

```
69.         end
70.         4:begin
71.             st <= 5;
72.         end
73.         5:begin
74.             st <= 1;
75.             start <= 1'b1;
76.         end
77.         default : st <= 0;
78.         endcase
79.
80.     end
81.
82.
83.     acc  #
84.     (
85.         .width(width),
86.         .len(len)
87.     )
88.     acc_inst
89.     (
90.         .rst(rst),
91.         .clk(clk),
92.         .start(start),
93.         .data_rdy(data_rdy),
94.         .indata(indata),
95.
96.         .outdata(outdata),
97.         .cal_done(cal_done)
98.     );
99.
100.
101.    endmodule
```

仿真结果如图2所示：

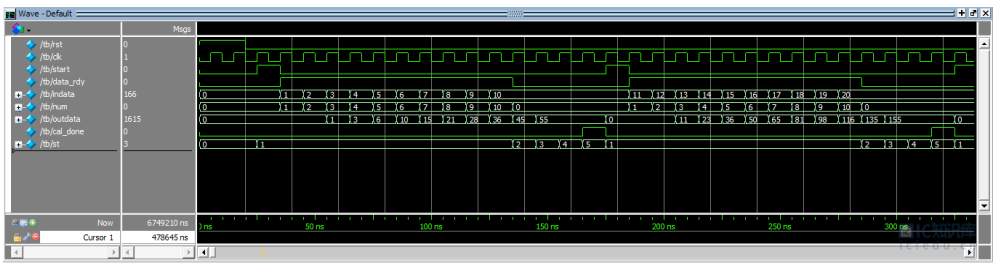

图2

从图2的波形可以看出仿真结果与设计意图相符。

对应视频：

- Verilog 全加器，全减器，累加器设计–视频教程-1
- Verilog 全加器，全减器，累加器设计–视频教程-2
- Verilog 全加器，全减器，累加器设计–视频教程-3

Posted in FPGA, FPGA开发板, Verilog, Verilog

Tagged Verilog accumulator, 累加器设计，全加器设计

发表回复

以 Byron 的身份登录。 编辑您的个人资料。 注销？ 必填项已用*标注

评论 *

如果有人回复我的评论，请通过电子邮件通知我。

发表评论

相关链接

系统验证基础–OVM与UVM

无线信号建模–多径效应

CPU cache设计由来

浮点运算计算方法 (乘除法)

浮点运算计算方法 (加减法)

FPGA 浮点运算算法

I2C cmd层 Verilog程序结构化设计

I2C 主设备PHY层STOP控制Verilog程序设计

I2C 主设备PHY层读控制Verilog程序设计

I2C 主设备PHY层写控制Verilog程序设计

I2C PHY层 Verilog程序结构化设计

I2c 协议实现EEPROM 仿真

今日热门文章

- Python 查找list平均值
- Verilog 文件操作-$fgetc，$fgets，$fscanf，$fread
- Verilog 文件操作-$fdisplay，$fwrite，$fstrobe，$fmonitor
- 芯片天地-网络视频学院（视频）
- 电流镜 (Current Mirror)
- System Verilog 教材
- Verilog语言编程与FPGA应用教材
- RISC-V CSR寄存器（2）CSR寄存器
- System Verilog 类的定义与使用（一）
- RISC-V CSR寄存器（1）CSR简介和CSR指令
- Verilog仿真中fork…join的原理和使用
- Verilog 移位操作符
- RISC-V 教学教案
- Verilog case语句
- Verilog 全加（减）器及累加器设计

近期评论

本月热门文章

- Verilog语言编程与FPGA应用教材
- FPGA与Verilog 教学视频
- RISC-V 教学教案
- Python 查找list平均值
- 芯片天地-教材与教案（文章）
- 芯片天地-网络视频学院（视频）
- Verilog 文件操作-$fdisplay，$fwrite，$fstrobe，$fmonitor
- RISC-V CSR寄存器（1）CSR简介和CSR指令
- Verilog 文件操作-$fgetc，$fgets，$fscanf，$fread
- 电流镜 (Current Mirror)
- Verilog 文件操作-$fopen，$fclose
- 第三节 卡诺图原理及构成
- Verilog 中not ~ ！的区别
- RISC-V 指令集手册中文2.1版（User-Level ISA）
- RISC-V指令集进解

本站热门文章

- Verilog语言编程与FPGA应用教材
- FPGA与Verilog 教学视频
- RISC-V 教学教案
- 芯片天地-教材与教案（文章）
- RISC-V 教学教案-视频
- 芯片天地-网络视频学院（视频）
- 电流镜 (Current Mirror)
- Python 查找list平均值
- Verilog 文件操作-$fgetc，$fgets，$fscanf，$fread
- Verilog 文件操作-$fdisplay，$fwrite，$fstrobe，$fmonitor
- RISC-V CSR寄存器（1）CSR简介和CSR指令
- RISC-V 指令集手册中文2.1版（User-Level ISA）
- Verilog 文件操作-$fopen，$fclose
- Python time.sleep()为代码添加延迟
- SOC 教学教案

Copyright © 2023 芯片天地

RISC-V指令集讲解

（2）I-Type整数寄存器-立即数指令

Verilog 文件操作-$fseek，$ftell，$feof

I2C接口基础知识一

三级放大器的稳定性

SOC 教学教案

CMOS模拟集成电路设计教材

数字集成电路设计中的标准单元库介绍

I2C接口基础知识一