# 模块实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 叶福伦 | 学号 | 21051331 |
| 专业 | 计算机科学与技术 | 班级 | 21052313 |
| 指导教师 | 髙志刚 | 课程名称 | 计算机组成原理课程设计（甲） |
| 实验序号 | 02 | 实验名称 | 超前进位加法器设计实验 |
| 实验时间 | 05.05 | 实验地点 | 1教223 |

# 成绩表

|  |  |  |
| --- | --- | --- |
| **考核项目** | **考查分项** | **得分** |
| **一、实验方案设计（30分）** | **实验目的与要求（5分）** |  |
| **模块构成与连接（15分）** |  |
| **板级验证方案（10分）** |  |
| **二、FPGA程序设计（20分）** | **程序源代码（15分）** |  |
| **管脚约束程序源代码（5分）** |  |
| **三、仿真实验（15分）** | **仿真代码（5分）** |  |
| **仿真波形及结果分析（10分）** |  |
| **四、板级实验（15分）** | **实验操作（5分）** |  |
| **结果记录（5分）** |  |
| **结果分析（5分）** |  |
| **五、思考与探索（10分）** | **拓展提升（10分）** |  |
| **格式（10分）** | **书写规范、图表清晰、表达清楚** |  |
| **合计** | |  |

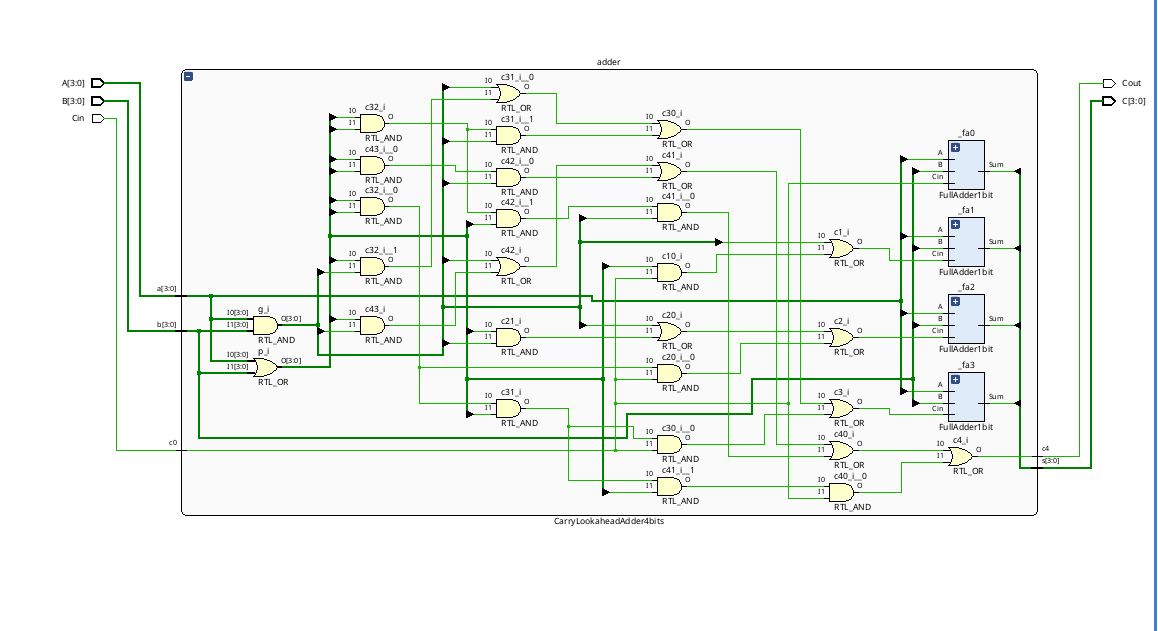
# 一、实验方案设计

1. 实验目的与要求（5分）

掌握全加器、串行进位加法器和超前进位加法器的原理与设计方法； 掌握运用Verilog HDL语言进行结构和数据流描述与建模方法； 掌握运用Verilog HDL语言进行模块调用的方法。

1. 模块构成与连接（15分）

（请附图，并说明各模块的功能，以及模块间的连接关系。也可附开发工具中显示的电路模块连接图，加以说明。）

左侧是超前进位电路，产生进位信号，产生的最高一级信号作为Cout输出其余作为FA0-3的进位信号参与全加法运算输出C

1. 板级验证方案（10分）

（说明顶层测试模块的输入/输出信号，及对应使用的I/O设备；可附图说明）

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 信号名 | 使用的IO设备 | 说明 |
| input | Cin | 逻辑开关 | 输入的进位信号 |
| A[3:0] | 一组逻辑开关 | 被加数 |
| B[3:0] | 一组逻辑开关 | 加数 |
| output | Cout | LED灯 | 产生的进位信号 |
| C[3:0] | 一组LED灯 | 结果 |

# 二、FPGA程序设计

1. 实验程序源代码（25分）

（实验各个模块的代码，包含功能注释）

//FullAdder1bit.v

module FullAdder1bit (

input A,

input B,

input Cin,

output Sum,

output Cout

);

wire AxorB, AandB, AxorBandCin;

xor (Sum, A, B, Cin);

and (AandB, A, B);

xor (AxorB, A, B);

and (AxorBandCin, AxorB, Cin);

or (Cout, AandB, AxorBandCin);

endmodule

//CarryLookaheadAdder4bits.v

`include "../01-serial-adder/FullAdder1bit.v"

module CarryLookaheadAdder4bits(

input [3:0] a,

input [3:0] b,

input c0,

output [3:0] s,

output c4

);

wire [3:0] g;

wire [3:0] p;

assign g = a & b;

assign p = a | b;

wire [4:0] c;

assign c1 = g[0] | (p[0] & c0);

assign c2 = g[1] | (p[1] & g[0]) | (p[0] & p[1] & c0);

assign c3 = g[2] | (p[2] & g[1]) | (p[1] & p[2] & g[0]) | (p[0] & p[1] & p[2] & c0);

assign c4 = g[3] | (p[3] & g[2]) | (p[2] & p[3] & g[1]) | (p[1] & p[2] & p[3] & g[0]) | (p[0] & p[1] & p[2] & p[3] & c0);

FullAdder1bit \_fa0(

.A(a[0]),

.B(b[0]),

.Cin(c0),

.Sum(s[0]),

.Cout()

);

FullAdder1bit \_fa1(

.A(a[1]),

.B(b[1]),

.Cin(c1),

.Sum(s[1]),

.Cout()

);

FullAdder1bit \_fa2(

.A(a[2]),

.B(b[2]),

.Cin(c2),

.Sum(s[2]),

.Cout()

);

FullAdder1bit \_fa3(

.A(a[3]),

.B(b[3]),

.Cin(c3),

.Sum(s[3]),

.Cout()

);

endmodule

//CarryLookaheadAdder4bits.top.v 顶层模块

module CarryLookaheadAdder4bitsTopper(

input [3:0] A,

input [3:0] B,

input Cin,

output [3:0] C,

output Cout

);

CarryLookaheadAdder4bits adder(A,B,Cin,C,Cout);

endmodule

1. FPGA管脚约束代码（5分）

（引脚约束文件的内容，备注使用的具体设备）

A，B 使用逻辑开关，input Cin使用逻辑开关，C[3:0]，Cout使用led灯

set\_property PACKAGE\_PIN R1 [get\_ports Cout]

set\_property PACKAGE\_PIN AB8 [get\_ports Cin]

set\_property PACKAGE\_PIN T3 [get\_ports {A[3]}]

set\_property PACKAGE\_PIN U3 [get\_ports {A[2]}]

set\_property PACKAGE\_PIN T4 [get\_ports {A[1]}]

set\_property PACKAGE\_PIN V3 [get\_ports {A[0]}]

set\_property PACKAGE\_PIN V4 [get\_ports {B[3]}]

set\_property PACKAGE\_PIN W4 [get\_ports {B[2]}]

set\_property PACKAGE\_PIN Y4 [get\_ports {B[1]}]

set\_property PACKAGE\_PIN Y6 [get\_ports {B[0]}]

set\_property PACKAGE\_PIN G1 [get\_ports {C[3]}]

set\_property PACKAGE\_PIN E1 [get\_ports {C[2]}]

set\_property PACKAGE\_PIN D2 [get\_ports {C[1]}]

set\_property PACKAGE\_PIN A1 [get\_ports {C[0]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports Cin]

set\_property IOSTANDARD LVCMOS18 [get\_ports Cout]

set\_property IOSTANDARD LVCMOS18 [get\_ports {A[3]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {A[2]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {A[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {A[0]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {B[3]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {B[2]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {B[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {B[0]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {C[3]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {C[2]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {C[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {C[0]}]

set\_property PULLDOWN true [get\_ports Cin]

set\_property PULLDOWN true [get\_ports {A[3]}]

set\_property PULLDOWN true [get\_ports {A[2]}]

set\_property PULLDOWN true [get\_ports {A[1]}]

set\_property PULLDOWN true [get\_ports {A[0]}]

set\_property PULLDOWN true [get\_ports {B[3]}]

set\_property PULLDOWN true [get\_ports {B[2]}]

set\_property PULLDOWN true [get\_ports {B[1]}]

set\_property PULLDOWN true [get\_ports {B[0]}]

set\_property BITSTREAM.GENERAL.COMPRESS TRUE [current\_design]

# 三、仿真实验

（只对CPU模块仿真，无需对板级验证的顶层测试模块仿真）

1. 仿真代码（5分）

（含仿真源代码、仿真验证方案）

`timescale 1ns / 1ps

module CarryLookaheadAdder4bitsTester();

reg [3:0]A;

reg [3:0]B;

reg Cin;

wire [3:0]C;

wire Cout;

CarryLookaheadAdder4bits adder(

A,

B,

Cin,

C,

Cout

);

initial begin

A = 4'b0000;

B = 4'b0000;

Cin = 0;

#100

A = 4'b0001;

B = 4'b0100;

Cin = 0;

#100

A = 4'b1111;

B = 4'b1111;

Cin = 1;

#100

A = 4'b1100;

B = 4'b0011;

Cin = 0;

#100

A = 4'b1101;

B = 4'b0011;

Cin = 1;

#1000

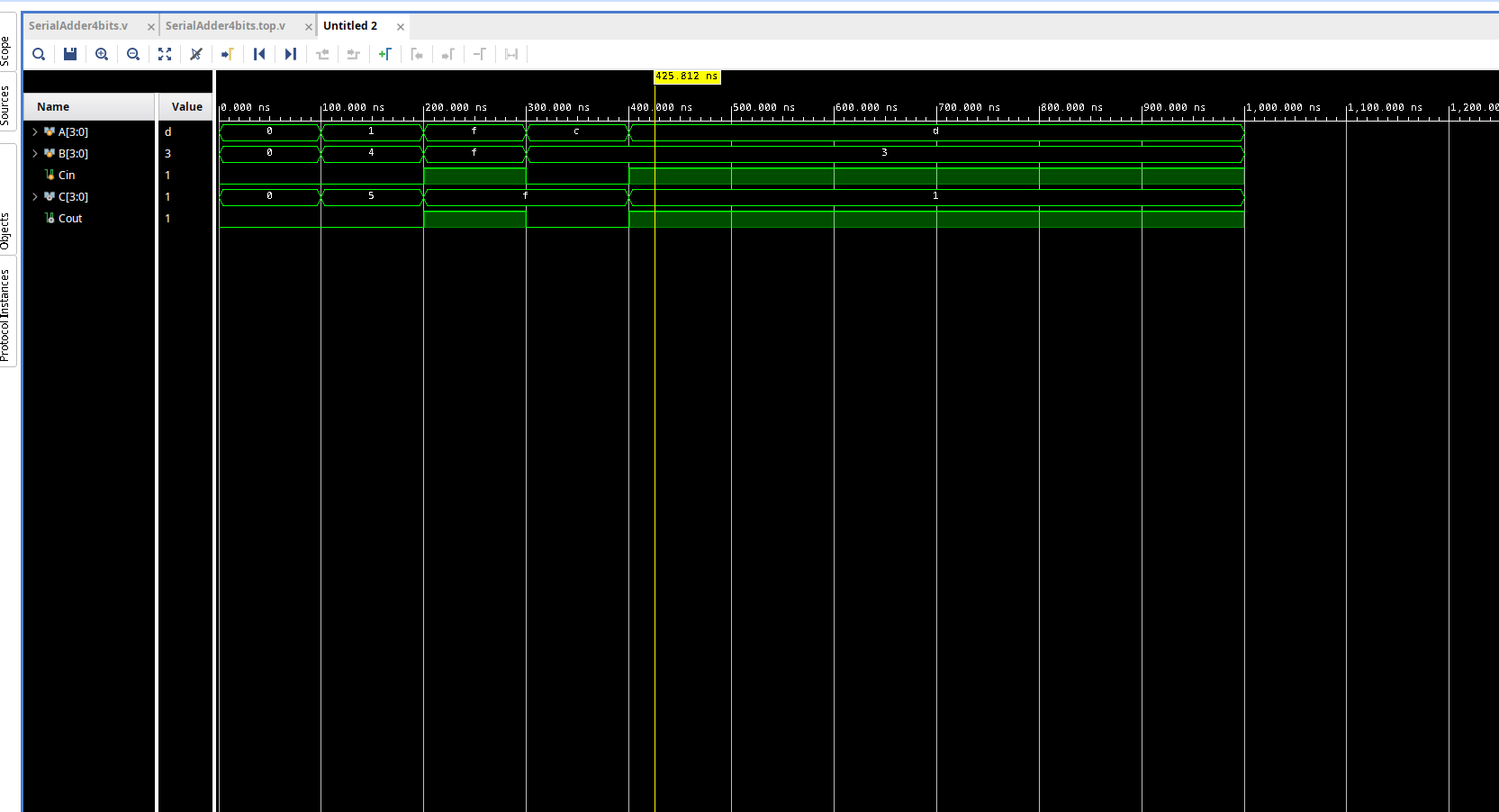
$finish;

end

endmodule

1. 仿真波形及结果分析（15分）

（仿真波形截图，并对仿真波形进行分析）



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 时间段 | A | B | Cin | C | 期望的C | Cout | 期望的Cout | 说明 |
| 000 - 100ns | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 考察没有cin时是否出错 |
| 100 - 200ns | 1 | 4 | 0 | 5 | 5 | 0 | 0 | 同上 |
| 200 - 300ns | F | F | 1 | F | F | 1 | 1 | 考察有cin且有进位的情况 |
| 300 - 400ns | C | 3 | 0 | F | F | 0 | 0 | 考察没有cin但是有进位的情况 |
| 400ns之后 | D | 3 | 1 | 1 | 1 | 1 | 1 |  |

# 通过仿真结果可以大致确定电路是正确的

# 四、板级实验

1. 板级实验操作说明（5分）

进行引脚配置，对4位二进制超前进位加法器模块进行板级验证。拨动输入信号A、B和C0所对应的逻辑开关，输入典型的数据，观察LED灯，记录分析实验结果。

1. 板级实验结果记录（5分）

（以表格形式展现，具体见教材上实验结果记录表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | C | Cout |
| 0000 | 0000 | 0 | 0000 | 0 |
| 1111 | 1111 | 1 | 1111 | 1 |
| 0000 | 0001 | 0 | 0001 | 0 |
| 1111 | 0000 | 1 | 0000 | 1 |
| 1110 | 1011 | 0 | 1001 | 1 |
| 1010 | 1011 | 1 | 0110 | 1 |
| 0001 | 1100 | 1 | 1110 | 0 |
| 0111 | 0011 | 0 | 1010 | 0 |

1. 结果分析与结论（5分）

（分析实验结果，给出实验结论）

(1) 由生成的电路图可知，FA之间的进位信号不存在依赖关系，进位信号统一由超前进位电路产生所以为超前进位加法器

(2) C和Cout的结果正确，虽然没有覆盖到所有可能性，但能判断超前进位加法器的正确性。

# 五、思考与探索

（10分）

1. 问题与解决方案

（整个实验过程中发生了什么问题？你是如何解决的。）

运算结果出错：中间产生g和p的电路出错，g应该是g = a & b而不是a | b;

p应该是a | b而不是a & b,纠正后结果就正确了

1. 思考题

（力所能及，尝试实践或回答教材上的思考与探索题目，至少完成1道）

1. 实验1是串行电路，实验2是超前进位电路，实验1电路简单但效率低实验2电路较为复杂但运算效率高

2. 实验一4个全加器的运算具有依赖性。而2中的全加器运算不存在先后依赖

3. 超前进位

# 六、实验心得体会、意见建议

（如果是小组合作，请给出具体分工、分别给出心得体会）

当我进行超前进位加法器的实验时，我发现它相比串行进位加法器有着更快的运算速度和更高的效率，但同时也需要更多的硬件资源来实现。

在实验中，我首先了解了超前进位加法器的原理和基本结构，然后根据实验要求进行了电路设 计和Verilog HDL编程。我发现，实验过程中需要注意以下几点：

1. 设计正确的电路结构，包括进位预测模块、进位生成模块和加法器模块。不同的电路结构可能会影响加法器的运算速度和精度，因此需要根据实际情况进行调整。
2. 确保Verilog HDL代码正确无误，包括正确实现了超前进位加法器的逻辑功能和正确连接各个模块。可以通过仿真验证代码的正确性。

通过这次实验，我更深入地了解了超前进位加法器的原理和设计方法，掌握了Verilog HDL的 基本使用方法，并对数字电路的设计和实现有了更深刻的认识。