课程报告

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** |  | **学号** |  |
| **实验时间** | 2023.12.1 | **实验地点** | 第1教研楼225 |
| **实验序号** | 实验2 | **实验名称** | 超前进位加法器设计 |
| **课程名称** | 数字电路课程设计 | **指导教师** | 潘玉剑 |

# 方案设计

包括实验目的、模块设计说明（基于个人的程序代码，只有一个模块时，说明模块module的输入输出功能；存在多个模块时，还要提供模块架构图/说明，即模块与模块之间的关系）、输入输出I/O设定（输入输出变量的I/O管脚需求分析，及对应的板级硬件资源配置理由，即如何选择和利用实板上的硬件资源，如变量对应按键、LED、数码管等），共30分

## 实验目的

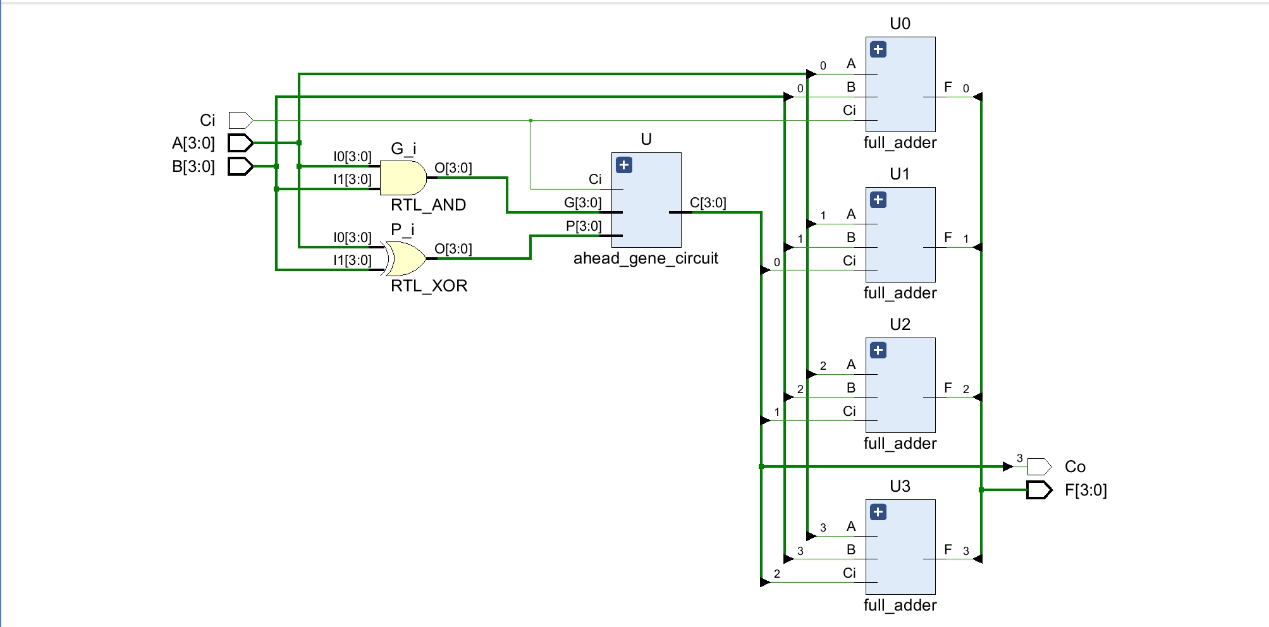
1.掌握超前进位加法器的原理与设计方法

2.掌握运用VerilogHDL语言进行数据流描述与建模方法

3.掌握运用VerilogHDL语音进行模块调用的方法

## 模块设计说明

代码包括三个模块，look\_ahead, ahead\_gene\_circuit 和 full\_adder 三个模块，其中look\_ahead为主模块，输入为要相加的四位二进制数A和B，还有上一位进位Ci，输出为相加后的四位二进制数F和下一位进位Co。look\_ahead调用两个子模块ahead\_gene\_circuit和full\_adder，ahead\_gene\_circuit模块并行计算每一位的进位和和数，输入为四位二进制数P=A^B和G=A&B，还有上一位进位Ci，输出为四位的进位数C。full\_adder模块是一位二进制数加法器，输入为要相加的一位二进制数A和B，还有进位Ci，输出为相加后的一位二进制数F。



## 输入输出I/O

输入的两个加数A，B和进位Ci对应板卡上的拨码开关，可以根据拨动开关的位置输入高或低电平至相应的FPGA引脚，向上拨输入1，向下拨输入0.输出的两数和F和下一位进位Co对应板卡上的输出设备LED显示灯，输出值为1时灯亮，输出值为0时灯灭。

# 逻辑程序设计

## 逻辑程序源代码

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2023/12/01 14:23:42

// Design Name:

// Module Name: look\_ahead

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module ahead\_gene\_circuit(

input [3:0] P,G,

input Ci,

output [3:0] C);

assign C[0]=G[0]|(P[0]&Ci);

assign C[1]=G[1]|(P[1]&C[0]);

assign C[2]=G[2]|(P[2]&C[1]);

assign C[3]=G[3]|(P[3]&C[2]);

endmodule

module look\_ahead(

input [3:0] A,B,

input Ci,

output [3:0] F,

output Co);

wire [3:0] P,G,C;

assign G=A&B;

assign P=A^B;

ahead\_gene\_circuit U(P,G,Ci,C);

full\_adder U0(A[0],B[0],Ci,F[0]);

full\_adder U1(A[1],B[1],C[0],F[1]);

full\_adder U2(A[2],B[2],C[1],F[2]);

full\_adder U3(A[3],B[3],C[2],F[3]);

assign Co=C[3];

endmodule

module full\_adder(

input A,B,Ci,

output F);

assign F = A^B^Ci;

endmodule

## 管脚约束程序源代码

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

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 W7 [get\_ports {B[3]}]

set\_property PACKAGE\_PIN Y8 [get\_ports {B[2]}]

set\_property PACKAGE\_PIN Y7 [get\_ports {B[1]}]

set\_property PACKAGE\_PIN T1 [get\_ports {B[0]}]

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 {F[3]}]

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

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

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 PACKAGE\_PIN R1 [get\_ports {F[3]}]

set\_property PACKAGE\_PIN P2 [get\_ports {F[2]}]

set\_property PACKAGE\_PIN P1 [get\_ports {F[1]}]

set\_property PACKAGE\_PIN N2 [get\_ports {F[0]}]

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

set\_property PACKAGE\_PIN Y1 [get\_ports Ci]

set\_property PACKAGE\_PIN G1 [get\_ports Co]

set\_property IOSTANDARD LVCMOS18 [get\_ports Ci]

set\_property IOSTANDARD LVCMOS18 [get\_ports Co]

set\_property PULLDOWN true [get\_ports Ci]

# 测试程序设计

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2023/12/01 14:40:30

// Design Name:

// Module Name: look\_ahead\_test

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

`timescale 10ns/1ns

module look\_ahead\_test();

reg [3:0] A,B;

reg Ci;

wire [3:0] S;

wire Co;

look\_ahead U(A,B,Ci,S,Co);

initial

$monitor($time,"\tA=%b,B=%b,Ci=%b,S=%b,Co=%b",A,B,Ci,S,Co);

initial begin

A=4'b0000;B=4'b0000;Ci=0;

#5;

A=4'b0000;B=4'b0001;Ci=0;

#5;

A=4'b0001;B=4'b0001;Ci=0;

#5;

A=4'b0001;B=4'b0010;Ci=0;

#5;

A=4'b00010;B=4'b0010;Ci=0;

#5;

A=4'b0010;B=4'b0011;Ci=0;

#5;

A=4'b0011;B=4'b0011;Ci=0;

#5;

A=4'b0100;B=4'b0011;Ci=0;

#5;

A=4'b0100;B=4'b0100;Ci=0;

#5;

A=4'b0100;B=4'b0101;Ci=0;

#5;

A=4'b0101;B=4'b0101;Ci=0;

#5;

A=4'b0110;B=4'b0101;Ci=0;

#5;

A=4'b0110;B=4'b0110;Ci=0;

#5;

A=4'b0111;B=4'b0110;Ci=0;

#5;

A=4'b0111;B=4'b0111;Ci=0;

#5;

A=4'b1000;B=4'b0111;Ci=0;

#5;

A=4'b1000;B=4'b1000;Ci=0;

#5;

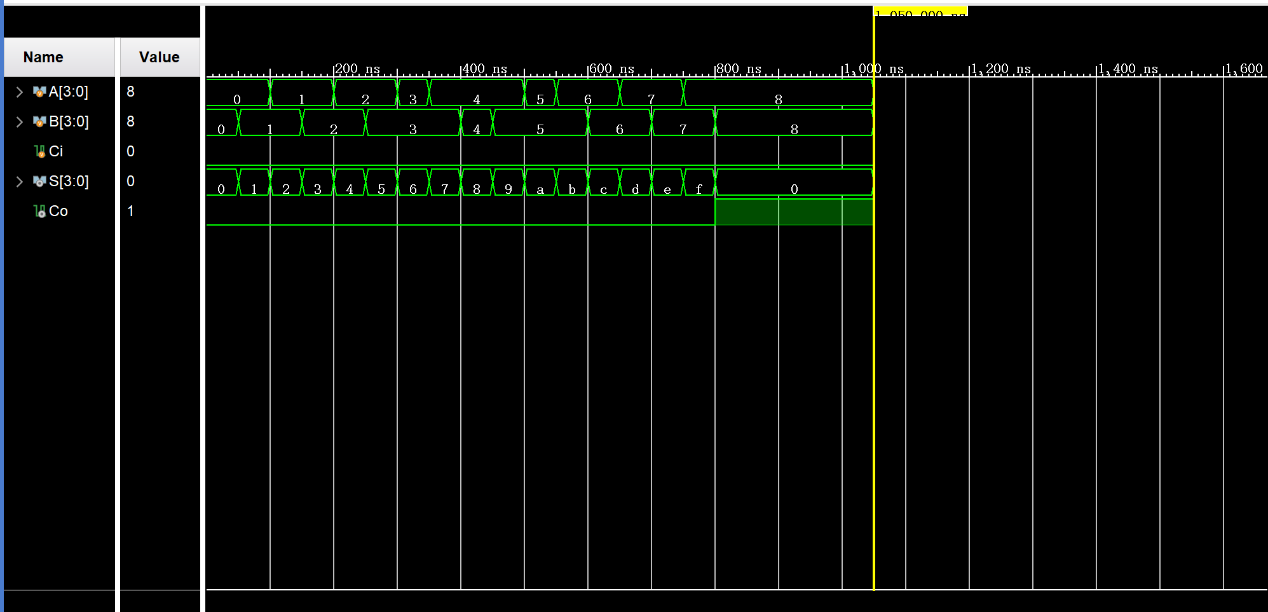
$stop;

end

endmodule

# 实验结果分析

## 仿真波形



## 仿真结果分析

50ns输入0和1，上一位进位0，输出1，进位0

100ns输入1和1，上一位进位0，输出2，进位0

400ns输入4和4，上一位进位0，输出8，进位0

600ns输入6和6，上一位进位0，输出12，进位1

800ns输入8和8，上一位进位0，输出0，进位1

## 实验结论

参加运算的数据A和B送到超前进位电路，然后由超前进位电路同时产生各位的进位C1，C2，C3，C4，然后再由各位全加器同时产生各位的和数。

## 实验出现的问题与解决方案

问题：如何把串行进位改为并行进位

解决方案：先把各位的进位算好，再用四个全加器分别计算各位的和数

# 思考与探索

1.请说明你本次实验实现的4位加法器，和实验1实现加法器的不同之处。

本次实验实现的4位加法器是并行进位加法器，即先把各位的进位算好，再用四个全加器并行计算各位的和数。实验1实现的加法器是串行进位加法器，采用的是先计算本位的和数和进位，再传送到下一位计算和数和进位。并行进位相比串行进位计算速度更快，效率更高。

# 意见和建议

**实验内容的建议：**

**可以提供不同级别和难度的设计要求，以满足不同学生的需求和能力水平。这有助于激发学生的兴趣，并使实验更具挑战性。**