

SHANGHAI JIAOTONG UNIVERSITY



计算机系统结构实验报告–Lab２

姓名：刘一凡

学号：518021910609

完成时间：2020/4/1７

目录

[1. 概述 3](#_Toc40977409)

[1.1 实验名称 3](#_Toc40977410)

[1.2 实验目的 3](#_Toc40977411)

[2. 实验描述 3](#_Toc40977412)

[2.1 adder\_4bits模块 3](#_Toc40977413)

[2.1.1 模块描述 3](#_Toc40977414)

[2.1.2 adder\_1bit模块代码 3](#_Toc40977415)

[2.1.3 adder\_4bits模块代码 4](#_Toc40977416)

[2.2 功能仿真 4](#_Toc40977417)

[2.2.1 步骤描述 4](#_Toc40977418)

[2.2.2 仿真代码 4](#_Toc40977419)

[2.2.3 仿真波形 5](#_Toc40977420)

[2.2.4 实验结论 6](#_Toc40977421)

[2.3 工程实现 6](#_Toc40977422)

[2.3.1 步骤描述 6](#_Toc40977423)

[2.3.2 顶层源文件代码 6](#_Toc40977424)

[2.3.3 约束代码 8](#_Toc40977425)

[3. 实验心得 9](#_Toc40977426)

# 1. 概述

## 1.1 实验名称

FPGA 基础实验：4-bit Adder

## 1.2 实验目的

1) 掌握Xilinx逻辑设计工具Vivado的基本操作

2) 掌握VerilogHDL进行简单的逻辑设计

3) 使用功能仿真；

4) 约束文件的使用和直接写法

5) 添加时序约束

6) 生成Bitstream 文件

7) 上板验证（暂定不做）

# 2. 实验描述

## 2.1 adder\_4bits模块

### 2.1.1 模块描述

这一模块由adder\_1bit模块和adder\_4bit 模块组成。adder\_1bit模块做1位的加法。adder\_4bits模块利用adder\_1bit模块实现四位的加法器。

### 2.1.2 adder\_1bit模块代码

1. module adder\_1bit(
2. input a,
3. input b,
4. input ci,
5. output s,
6. output co
7. );
8. wire s1,c1,c2,c3;
9. and (c1,a,b),
10. (c2,b,ci),
11. (c3,a,ci);
12. xor (s1,a,b),
13. (s,s1,ci);
14. or  (co,c1,c2,c3);
15. endmodule

### 2.1.3 adder\_4bits模块代码

1. module adder\_4bits(
2. input [3:0] a,
3. input [3:0] b,
4. input ci,
5. output [3:0] s,
6. output co
7. );
8. wire [2:0] ct;
10. adder\_1bit a1(.a(a[0]), .b(b[0]), .ci(ci), .s(s[0]), .co(ct[0])),
11. a2(.a(a[1]), .b(b[1]), .ci(ct[0]), .s(s[1]), .co(ct[1])),
12. a3(.a(a[2]), .b(b[2]), .ci(ct[1]), .s(s[2]), .co(ct[2])),
13. a4(.a(a[3]), .b(b[3]), .ci(ct[2]), .s(s[3]), .co(co));
14. endmodule

## 2.2 功能仿真

### 2.2.1 步骤描述

此功能仿真通过添加加法器的仿真激励文件，在模拟窗口用波形图显示加法器的运算结果，并可以验证程序是否正确。

### 2.2.2 仿真代码

1. module adder\_4bits\_tb(
3. );
5. reg [3:0] a;
6. reg [3:0] b;
7. reg ci;
9. wire [3:0] s;
10. wire co;
12. adder\_4bits u0(
13. .a(a),
14. .b(b),
15. .ci(ci),
16. .s(s),
17. .co(co));
19. initial begin
20. a=0;
21. b=0;
22. ci=0;
23. #100;
24. a=4'b0001;
25. b=4'b0010;
26. #100;
27. a=4'b0010;
28. b=4'b0100;
30. #100;
31. a=4'b1111;
32. b=4'b0001;
33. #100;
34. ci=1'b1;
35. end
36. endmodule

### 2.2.3 仿真波形

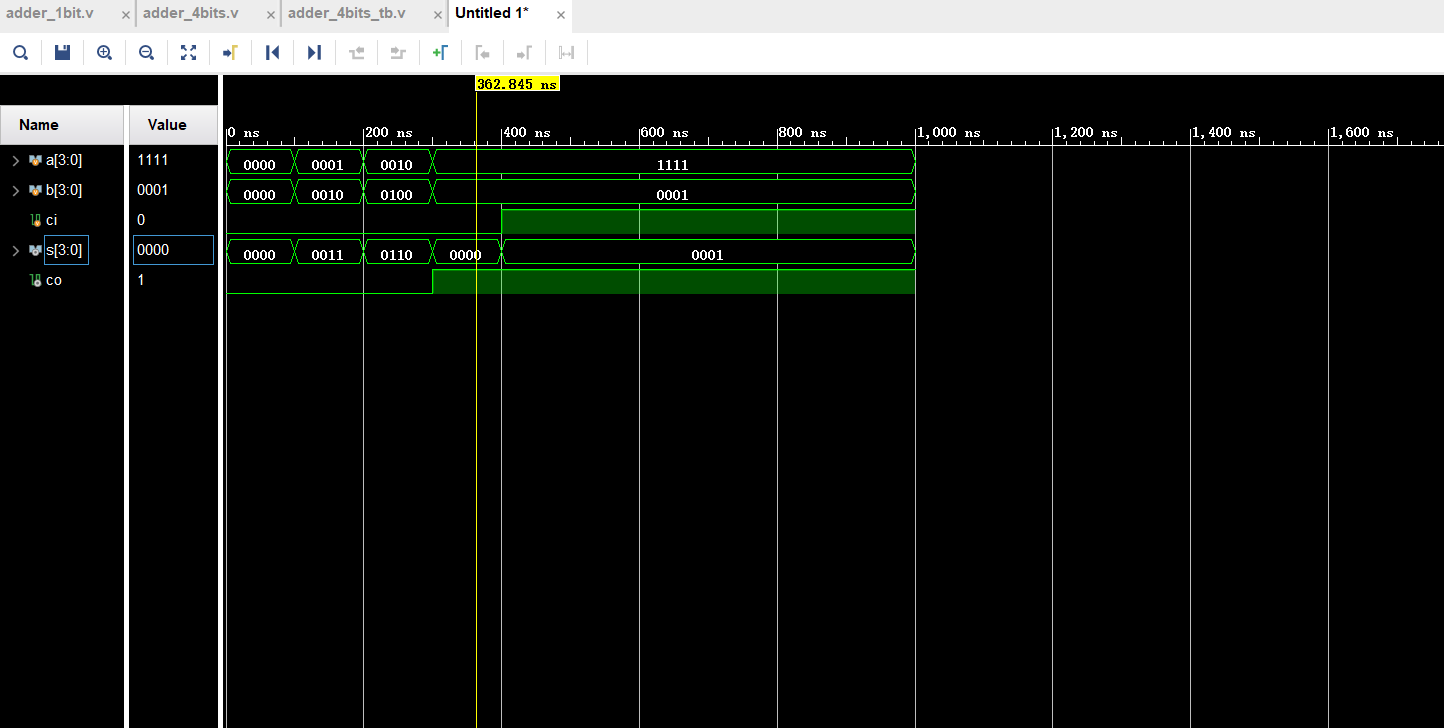


图1：四位加法器的仿真波形

### 2.2.4 实验结论

观察波形图，可知仿真结果与逻辑功能一致，电路能正常工作。例如，在两数位0001和0010时，相加结果为0011，结果正确。说明此四位加法器能够正确运行并得到合适的波形图，实验成功。

## 2.3 工程实现

### 2.3.1 步骤描述

在本次实验中是要用实验板上的8个Switch对应二组4位二进制输入，用4个LED发光二极管对应输出并用2个七段数码管显示运行结果。故本实验需要用display.v 这个七段数码管SEGMENT和LED 显示模块。需要创建一个顶层源文件和一个约束文件，并且在顶层文件中添加display IP核。

### 2.3.2 顶层源文件代码

1. module Top(
2. input clk\_p,
3. input clk\_n,
4. input [3:0] a,
5. input [3:0] b,
6. input reset,
8. output led\_clk,
9. output led\_do,
10. output led\_en,
12. output wire seg\_clk,
13. output wire seg\_en,
14. output wire seg\_do
15. );
17. wire CLK\_i;
18. wire Clk\_25M;
20. IBUFGDS IBUFGDS\_inst(
21. .O(CLK\_i),
22. .I(clk\_p),
23. .IB(clk\_n)
24. );
26. wire [3:0] s;
27. wire co;
28. wire [4:0] sum;
29. assign sum={co,s};
31. adder\_4bits U1(
32. .a(a),
33. .b(b),
34. .ci(1'b0),
35. .s(s),
36. .co(co)
37. );
39. reg [1:0] clkdiv;
40. always @ (posedge CLK\_i)
41. clkdiv<=clkdiv+1;
42. assign Clk\_25M=clkdiv[1];
44. display DISPLAY (
45. .clk(Clk\_25M),
46. .rst(1'b0),
47. .en(8'b00000011),
48. .data({27'b0, sum}),
49. .dot(8'b00000000),
50. .led(~{11'b0, sum}),
51. .led\_clk(led\_clk),
52. .led\_en(led\_en),
53. .led\_do(led\_do),
54. .seg\_clk(seg\_clk),
55. .seg\_en(seg\_en),
56. .seg\_do(seg\_do) );
58. endmodule

### 2.3.3 约束代码

1. set\_property PACKAGE\_PIN AC18 [get\_ports clk\_p]
2. set\_property IOSTANDARD LVDS [get\_ports clk\_p]
3. set\_property PACKAGE\_PIN AA12 [get\_ports {a[3]}]
4. set\_property PACKAGE\_PIN AA13 [get\_ports {a[2]}]
5. set\_property PACKAGE\_PIN AB10 [get\_ports {a[1]}]
6. set\_property PACKAGE\_PIN AA10 [get\_ports {a[0]}]
7. set\_property IOSTANDARD LVCMOS15 [get\_ports {a[0]}]
8. set\_property IOSTANDARD LVCMOS15 [get\_ports {a[1]}]
9. set\_property IOSTANDARD LVCMOS15 [get\_ports {a[2]}]
10. set\_property IOSTANDARD LVCMOS15 [get\_ports {a[3]}]
12. set\_property PACKAGE\_PIN AD10 [get\_ports {b[3]}]
13. set\_property PACKAGE\_PIN AD11 [get\_ports {b[2]}]
14. set\_property PACKAGE\_PIN Y12 [get\_ports {b[1]}]
15. set\_property PACKAGE\_PIN Y13 [get\_ports {b[0]}]
16. set\_property IOSTANDARD LVCMOS15 [get\_ports {b[0]}]
17. set\_property IOSTANDARD LVCMOS15 [get\_ports {b[1]}]
18. set\_property IOSTANDARD LVCMOS15 [get\_ports {b[2]}]
19. set\_property IOSTANDARD LVCMOS15 [get\_ports {b[3]}]
20. set\_property PACKAGE\_PIN N26 [get\_ports led\_clk]
21. set\_property PACKAGE\_PIN M26 [get\_ports led\_do]
22. set\_property PACKAGE\_PIN P18 [get\_ports led\_en]
23. set\_property IOSTANDARD LVCMOS33 [get\_ports led\_clk]
24. set\_property IOSTANDARD LVCMOS33 [get\_ports led\_do]
25. set\_property IOSTANDARD LVCMOS33 [get\_ports led\_en]
26. set\_property PACKAGE\_PIN M24 [get\_ports seg\_clk]
27. set\_property PACKAGE\_PIN L24 [get\_ports seg\_do]
28. set\_property PACKAGE\_PIN R18 [get\_ports seg\_en]
29. set\_property IOSTANDARD LVCMOS33 [get\_ports seg\_clk]
30. set\_property IOSTANDARD LVCMOS33 [get\_ports seg\_do]
31. set\_property IOSTANDARD LVCMOS33 [get\_ports seg\_en]

# 3. 实验心得

本次实验实现了四位加法器并进行了加法器的仿真模拟。在写源文件的过程中，我对于加法器的工作原理有了更深入的认识。加法器的最基本的单元是1位加法器，在此基础上将1位加法器进行组合就得到了我们需要的4位加法器。在写激励文件的过程中，我对于Verilog语法有了更多的理解，自己可以根据需要写对应的激励文件了，并且会通过“#+时间”语句来控制某信号延续的时长，波形仿真结果与逻辑上的运算结果完全一致，实验成功。

在写顶层源文件的过程中，我了解了IP核及其作用，并掌握了如何关联IP核以及源文件。在这次实验中，我对于软件和Verilog语言有了更深入的理解和认识。