浙江大学
ZHEJIANG UNIVERSITY

数字逻辑设计实验

实验8、全加器的设计实现

2023年11月

提 纲

- 实验目的
- 实验设备与材料
- 实验任务
- 实验原理
- 实验内容与步骤

实验目的

- 掌握一位全加器的工作原理和逻辑功能
- 掌握串行进位加法器的工作原理和进位延迟
- 了解加法器在CPU中的地位
- 掌握FPGA开发平台进行简单的I/O数据交互

实验设备与材料

- 实验设备
  - 装有Xilinx ISE 14.7的计算机　　1台
  - SWORD开发板　　1套
- 实验材料
  - 无

实验任务

- 任务1：设计4位串行进位加法器
- 任务2：实现4位加法器应用

实验原理

- 1位全加器
  - 三个输入位：数据位 A_i 和 B_i，低位进位输入 C_i
  - 二个输出位：全加和 S_i，进位输出 C_{i+1}

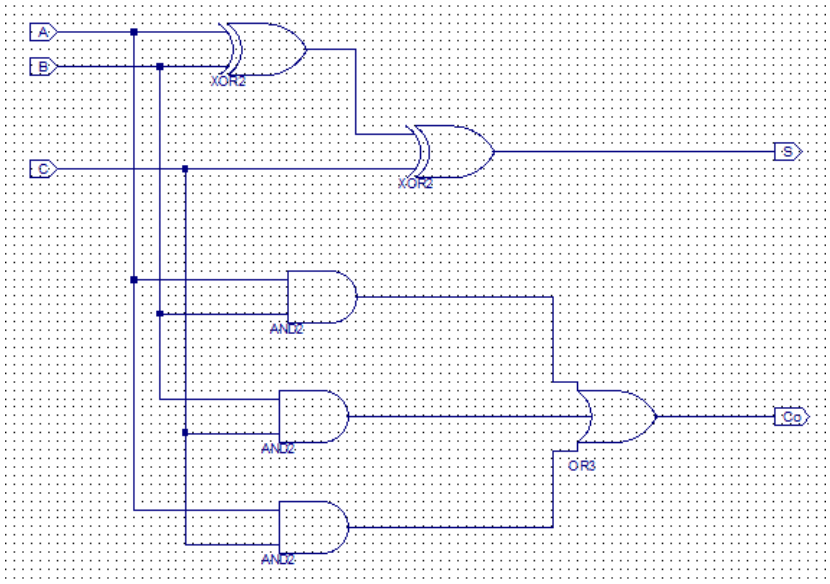
A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S_i = A_i \otimes B_i \otimes C_i$$

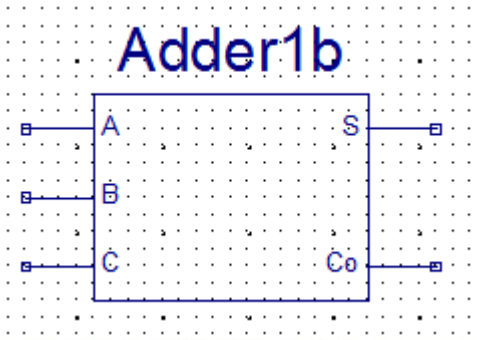
$$C_{i+1} = A_iB_i + B_iC_i + C_iA_i$$

一位全加器

- 根据一位全加器的输入输出关系，得到电路图

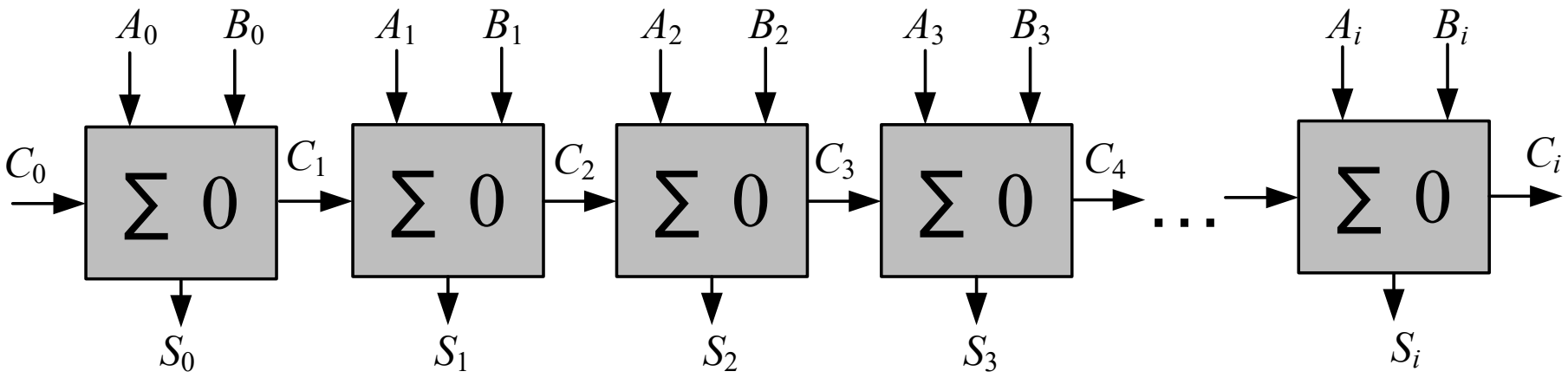



```
module adder_1bit(
    input wire a, b, ci,
    output wire s, co);
    // 根据真值表或等式填写
    // 对应的代码
endmodule
```

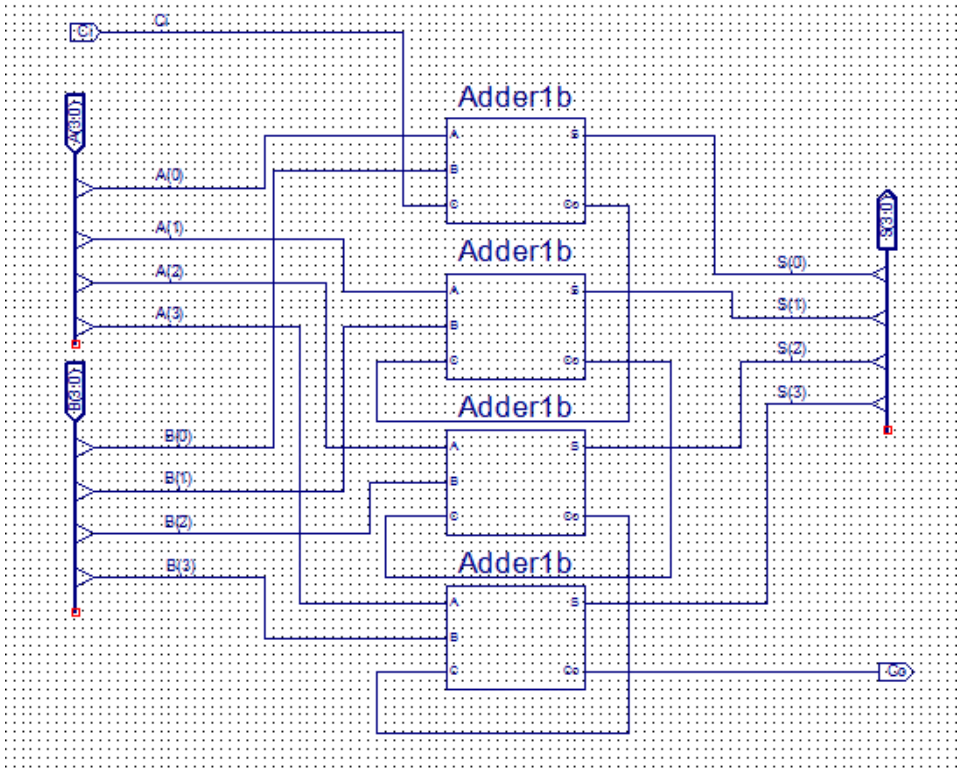


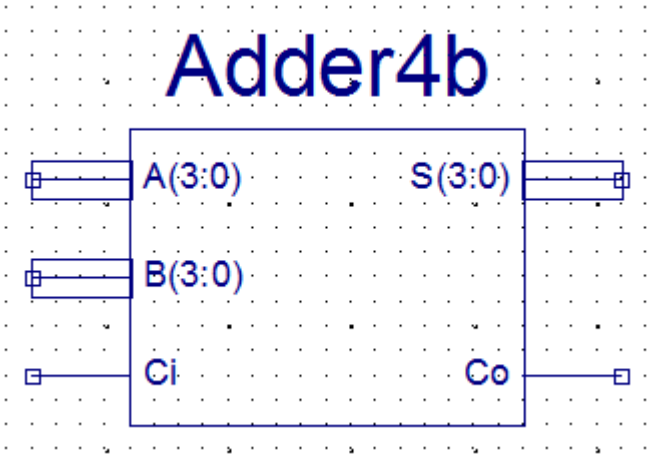
多位串行进位加法器

- 多位全加器可由一位全加器将进位串接构成
- 高位进位生成速度慢，位数越多时间越长



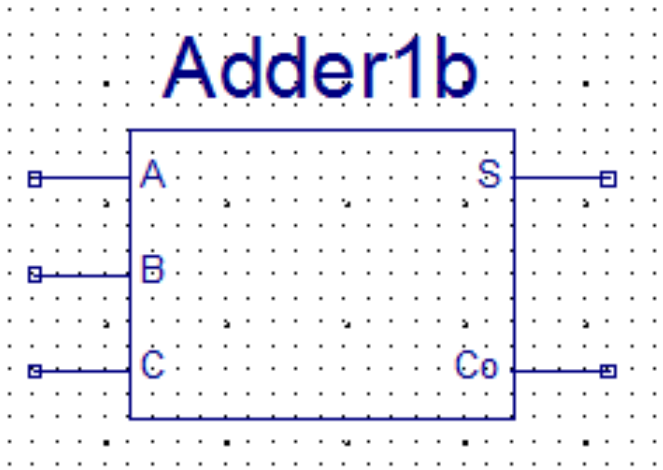
4位全加器


Ci
A(3:0)
A(0)
A(1)
A(2)
A(3)
B(3:0)
B(0)
B(1)
B(2)
B(3)
Adder1b
Adder1b
Adder1b
Adder1b
S(3:0)
S(0)
S(1)
S(2)
S(3)
Co


Adder4b
A(3:0)
B(3:0)
Ci
S(3:0)
Co

4位全加器(结构化描述)

```
module adder_4bit(
        input [3:0]A,
        input [3:0]B,
        input Ci,
        output [3:0]S,
        output Co
 );
        wire [0:2] Temp;
        // 根据电路图连接组件
endmodule
```



调用其他模型模版
调用了module模版（如Adder1b），调用的模块被命名为module_name（如add1），其中port_n端口连接wire_n线（如S端口连接wire Sum）
module module_name(.port_1(wire_1), … , .port_n(wire_n));

4位全加器(行为描述)

```
module adder_4bit(
        input [3:0]A,
        input [3:0]B,
        input Ci,
        output [3:0]S,
        output Co
    );
    wire [4:0] Temp;
    assign Temp=?;
    assign Co=?;
    assign S=;
endmodule
```

直接对组件的行为进行描述（类似于C语言）

防抖动模块

```
module pbdebounce(
   input wire clk_1ms,
   input wire button,
   output reg pbreg
 );
   // 设置寄存器（可以保存值不变直到再次被改变
   reg[7:0] pbshift;
   // 时序电路，在时钟上沿时触发
   always@(posedge clk_1ms) begin
      // 将8位都向左移动一位
      // 7 6 5 4 3 2 1 0
      // 6 5 4 3 2 1 0 0(注意，其他的数字代表位置，但是这个0代表补位为0)
      pbshift=pbshift<<1;
      // 将输入的值作为最后一位
      pbshift[0]=button;
      if(pbshift==8'b0)
         pbreg=0;
      // 为什么这样子可以防抖动
      if(pbshift==8'hFF)
         pbreg=1;
   end
endmodule
```

设计按键数据输入模块

- 使用行为描述设计
  - 在实验7基础上，更新“加法”为Adder4b模块

```
module CreateNumber(
input wire clk,
input wire [1:0] btn,
output reg [7:0] num
    );

wire [3:0] A,B;
wire [1:0] temp_btn;

initial num<=8'b 00010010;

assign A=num[3:0]+1'b1;
assign B=num[7:4]+1'b1;

pbdebounce p0(.clk_1ms(clk),.button(btn[0]),.pbreg(temp_btn[0]));
pbdebounce p1(.clk_1ms(clk),.button(btn[1]),.pbreg(temp_btn[1]));

always@(posedge temp_btn[0]) num[3:0]<=A;
always@(posedge temp_btn[1]) num[7:4]<=B;


endmodule
```

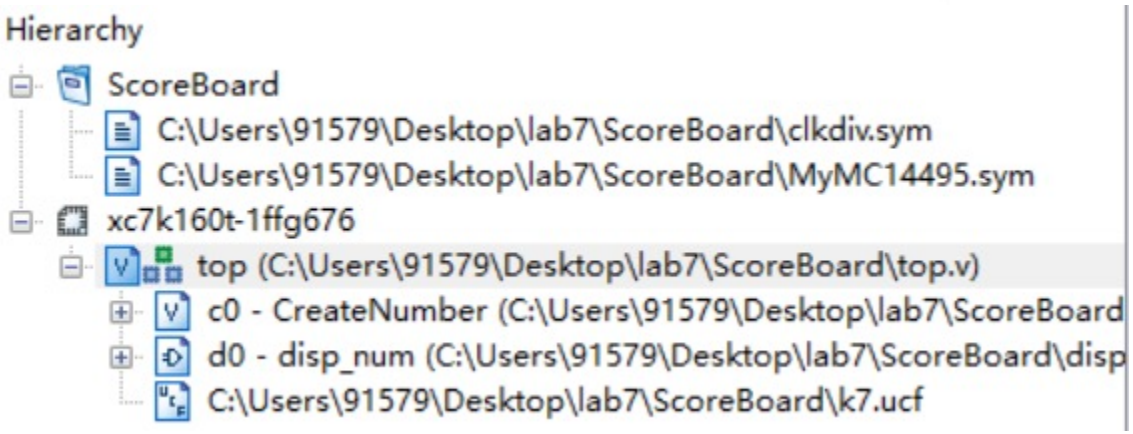
CreateNumber模块参考代码

```
module CreateNumber(
    input wire clk,
    input wire [1:0]btn,
    output reg [7:0]num,
    output C1,
    output [3:0]sum//num[3:0]和num[7:4]的和
    );

    wire [31:0] div;
    clkdiv cd0(.clkdiv(div),.clk(clk),.rst(1'b0));

    wire[3:0]A, B;
    wire A1,B1;
    wire[1:0]temp_btn;

    initial num <= 0;
    //to fill sth here
    adder_4bit a0(...);//A=num[3:0]+1,进位输出接wire A1
    adder_4bit a1(...);//B=num[7:4]+1,进位输出接Wire B2
    adder_4bit a2(...);//sum=num[3:0]+num[7:4],进位输出接output C1,Sum输出接output sum

    pbdebounce p0(.clk_1ms(div[17]),.button(btn[0]),.pbreg(temp_btn[0]));
    pbdebounce p1(.clk_1ms(div[17]),.button(btn[1]),.pbreg(temp_btn[1]));

    always@(posedge temp_btn[0]) num[3:0] <=A;
    always@(posedge temp_btn[1]) num[7:4] <=B;

endmodule
```

设计Top模块

```
module top(
            input wire clk,
            input wire [7:0]SW,
            input wire [1:0]btn,
            output wire [3:0]AN,
            output wire [7:0]SEGMENT
   );
             wire [7:0]num;
             wire [3:0]sum;
             wire C1;
             CreateNumber c0(.btn(btn), .num(num),
             .clk(clk),
             .sum(sum),
             .C1(C1));
             disp_num d0(.clk(clk),.HEXS({3'b0,C1,sum,num}),.LES(SW[7:4]),
             .points(SW[3:0]),.RST(1'b0),.AN(AN),
             .LED(SEGMENT[6:0]),.point(SEGMENT[7]));
endmodule
```

工程的整体结构


Hierarchy
ScoreBoard
C:\Users\91579\Desktop\lab7\ScoreBoard\clkdiv.sym
C:\Users\91579\Desktop\lab7\ScoreBoard\MyMC14495.sym
xc7k160t-1ffg676
top (C:\Users\91579\Desktop\lab7\ScoreBoard\top.v)
c0 - CreateNumber (C:\Users\91579\Desktop\lab7\ScoreBoard
d0 - disp_num (C:\Users\91579\Desktop\lab7\ScoreBoard\disp
C:\Users\91579\Desktop\lab7\ScoreBoard\k7.ucf

实验内容与步骤

- 任务1：设计4位串行进位加法器
- 任务2：实现4位加法器应用

4位加法器设计（schematic）

- 新建工程
  - 工程名称用MyAdder。
  - Top Level Source Type用HDL
- 新建源文件
  - 类型是Schematic
  - 文件名称用Adder1b。
- 原理图方式进行设计

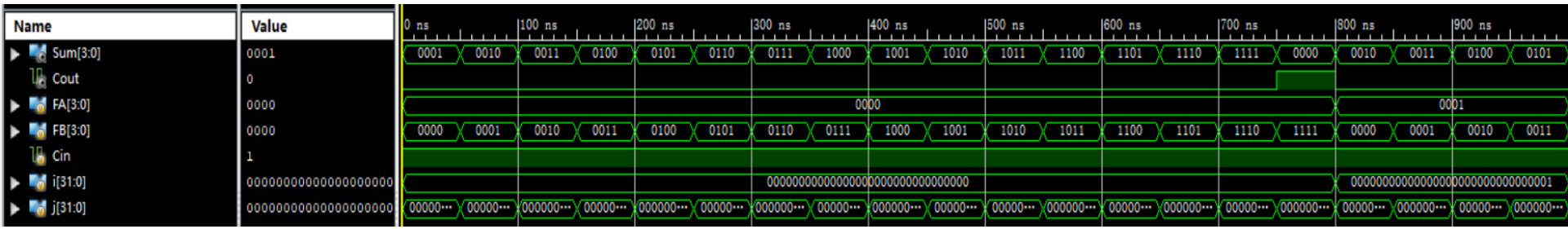
4位加法器设计（schematic）

- 新建源文件
  - 类型是Schematic
  - 文件名称用Adder4b。
- 原理图方式进行设计，调用前面设计的Adder1b
- 进行波形仿真

4位加法器设计（verilog）

- 下载工程
- 补全源文件
  - 根据注释和ppt提示补全adder_1bit.v和adder_4bit.v代码（根据原理图）
  - 根据注释和ppt提示补全adder_4bit.v代码（根据行为描述）

4位加法器设计（2）




```
initial begin
        Cin = 1'b1;
        for(i = 0; i < 16; i = i + 1) begin
                for(j = 0; j < 16; j = j + 1) begin
                        FA = i;
                        FB = j;
                        #50;
                end
        end
end
```

加法器应用设计（1）

- 步骤：
  - 1. 采用Verilog设计或电路图实现加法器模块
  - 2. 采用Verilog设计实现防抖动模块；
  - 3. 更新CreateNumber模块，加入防抖动模块并调用设计的加法器；
  - 4. 组合CreateNumber模块以及Lab7中的disp_num模块实现top模块；

加法器应用设计（2）

- 业务逻辑要求
  - 两个4位操作数A，B
  - 可用两个按键进行自增
  - 得到和C和进位Co
  - 把A、B、C和Co动态显示
  - 按键做防抖动处理

物理验证

```
NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18 ;

NET"btn[0]" LOC = AF10 | IOSTANDARD = LVCMOS15;
NET "btn[0]" clock_dedicated_route = false;
NET"btn[1]" LOC = AF13 | IOSTANDARD = LVCMOS15;
NET "btn[1]" clock_dedicated_route = false;

NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33 ;
NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33 ;
NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33 ;
NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33 ;

NET "SEGMENT[0]"          LOC = AB22  | IOSTANDARD = LVCMOS33 ;#a
NET "SEGMENT[1]"          LOC = AD24  | IOSTANDARD = LVCMOS33 ;#b
NET "SEGMENT[2]"          LOC = AD23  | IOSTANDARD = LVCMOS33 ;
NET "SEGMENT[3]"          LOC = Y21   | IOSTANDARD = LVCMOS33 ;
NET "SEGMENT[4]"          LOC = W20   | IOSTANDARD = LVCMOS33 ;
NET "SEGMENT[5]"          LOC = AC24  | IOSTANDARD = LVCMOS33 ;
NET "SEGMENT[6]"          LOC = AC23  | IOSTANDARD = LVCMOS33 ;#g
NET "SEGMENT[7]"          LOC = AA22  | IOSTANDARD = LVCMOS33 ;#point

NET "SW[0]"   LOC = AA10  | IOSTANDARD = LVCMOS15 ; #point0
NET "SW[1]"   LOC = AB10  | IOSTANDARD = LVCMOS15 ; #point1
NET "SW[2]"   LOC = AA13  | IOSTANDARD = LVCMOS15 ; #point2
NET "SW[3]"   LOC = AA12  | IOSTANDARD = LVCMOS15 ; #point3
NET "SW[4]"   LOC = Y13   | IOSTANDARD = LVCMOS15 ; #AN0
NET "SW[5]"   LOC = Y12   | IOSTANDARD = LVCMOS15 ; #AN1
NET "SW[6]"   LOC = AD11  | IOSTANDARD = LVCMOS15 ; #AN2
NET "SW[7]"   LOC = AD10  | IOSTANDARD = LVCMOS15 ; #AN3
```

Thank You !