

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机组成与设计 |
| 姓 名： |  |
| 学 院： | 信息与电子工程学院 |
| 系： |  |
| 专 业： |  |
| 学 号： |  |
| 指导教师： | 屈民军 唐奕 |

年 月 日

**实验报告**

专业：

姓名：

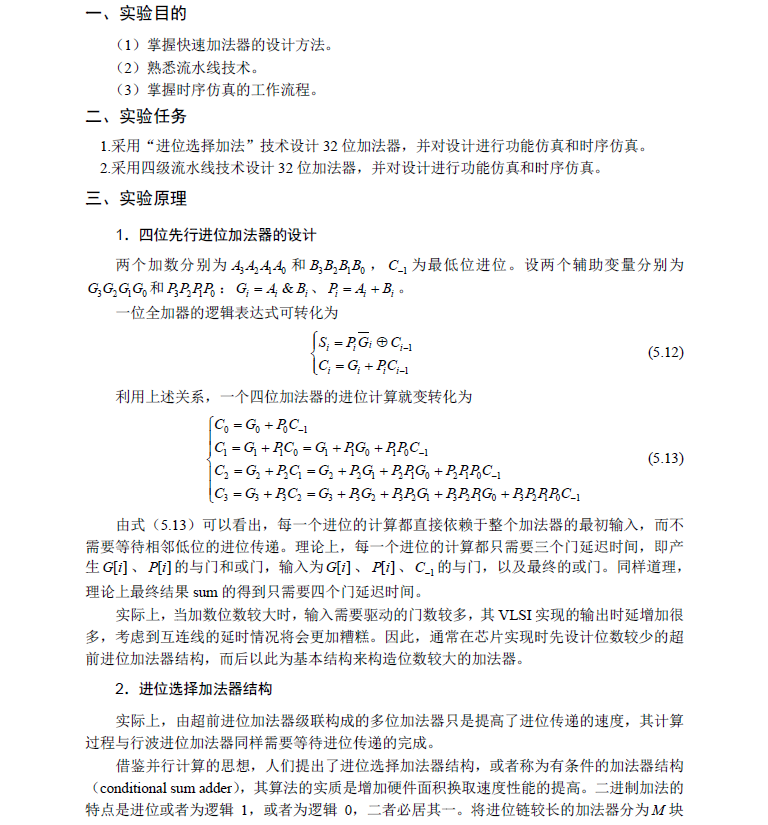
学号：

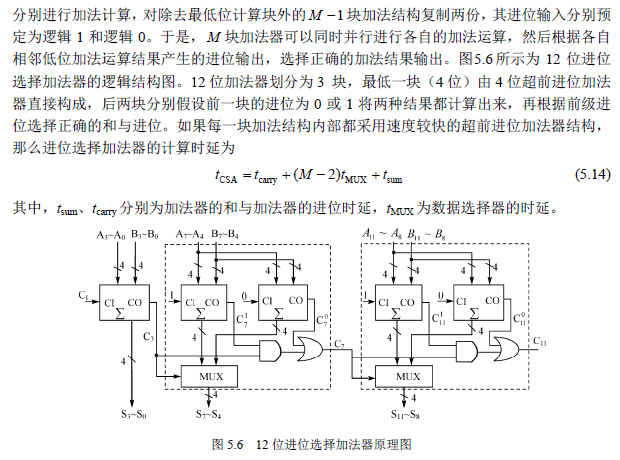
日期：

地点： 教十一400

课程名称：计算机组成与设计 指导老师：屈民军 唐奕 成绩：\_\_\_\_\_\_\_\_\_

实验名称：快速加法器的设计 实验类型：设计实验







**五、实验步骤与过程**

1. 编写4位先行进位加法器的Verilog HDL代码，并用ModelSim软件进行功能仿真。

（1）4位先行进位加法器的Verilog HDL代码

module adder\_4bits(a, b, ci, s, co);

    input[3:0] a, b;

    input ci;

    output[3:0] s;

    output co;

    wire[3:0] g, p;     // 辅助变量g和p，其中g[i] = a[i] & b[i]、 p[i] = a[i] + b[i]

    wire[2:0] c;        // 按教材C应当和加数A、B的位数保持一致，也就是4bit，但是实际上C[3]就是co，所以我们可以由 co c[2:0]  拼凑出 4bit的 C

    assign g = a & b;

    assign p = a | b;

    // 5.12(2) 先行进位计算

    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 co = g[3] | p[3]&c[2];

    // 5.12(1) 各位和计算

    assign s[0] = (p[0]&(~g[0])) ^ (ci);

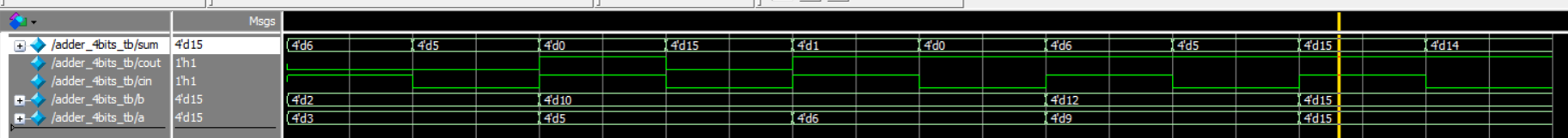
    assign s[1] = (p[1]&(~g[1])) ^ (c[0]);

    assign s[2] = (p[2]&(~g[2])) ^ (c[1]);

    assign s[3] = (p[3]&(~g[3])) ^ (c[2]);

endmodule

（2）ModelSim仿真



经过检验，加和(sum)和进位(cout)均无误。

备注：sum, b, a都调整为十进制无符号数(unsigned)格式显示，便于核验。cout,cin因为只有0,1两个取值所以十六进制也不妨碍观察。

2. 32位进位选择加法器的设计

（1）32位进位选择加法器的Verilog HDL代码

考虑到除第一级外，剩下7级加法器都会同时计算进位我0,1时的结果，并通过前一级的进位进行选择，作为最终该级加法器的输出。

故将七级加法器电路封装为adder\_4bits\_select模块，代码如下：

module adder\_4bits\_select (a, b, ci, s, co);

    input[3:0] a, b;

    input ci;

    output[3:0] s;

    output co;

    wire[3:0] s0, s1;

    wire c0, c1;

    adder\_4bits adder0(.a(a), .b(b), .ci(0), .s(s0), .co(c0));

    adder\_4bits adder1(.a(a), .b(b), .ci(1), .s(s1), .co(c1));

    // 根据是否有前级进位ci选择输出

    assign s = ci?s1:s0;

    assign co = ci?c1:c0;

endmodule

32位进位选择加法器的Verilog HDL代码

module adder\_32bits (a, b, ci, s, co);

    input[31:0] a, b;

    input ci;

    output[31:0] s;

    output co;

    wire c1, c2, c3, c4, c5, c6;

    adder\_4bits         adder0(.a(a[3:0]),      .b(b[3:0]),     .ci(ci), .s(s[3:0]), .co(c1));

    adder\_4bits\_select  adder1(.a(a[7:4]),      .b(b[7:4]),     .ci(c1), .s(s[7:4]), .co(c2));

    adder\_4bits\_select  adder2(.a(a[11:8]),     .b(b[11:8]),    .ci(c2), .s(s[11:8]), .co(c3));

    adder\_4bits\_select  adder3(.a(a[15:12]),    .b(b[15:12]),   .ci(c3), .s(s[15:12]), .co(c4));

    adder\_4bits\_select  adder4(.a(a[19:16]),    .b(b[19:16]),   .ci(c4), .s(s[19:16]), .co(c5));

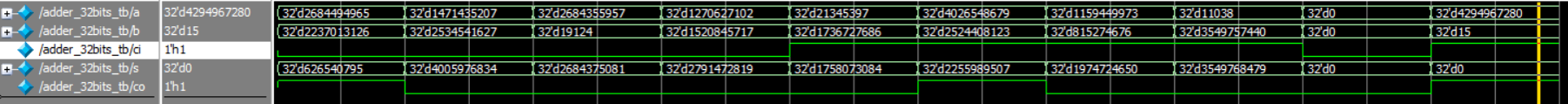
    adder\_4bits\_select  adder5(.a(a[23:20]),    .b(b[23:20]),   .ci(c5), .s(s[23:20]), .co(c6));

    adder\_4bits\_select  adder6(.a(a[27:24]),    .b(b[27:24]),   .ci(c6), .s(s[27:24]), .co(c7));

    adder\_4bits\_select  adder7(.a(a[31:28]),    .b(b[31:28]),   .ci(c7), .s(s[31:28]), .co(co));

endmodule

（2）ModelSim仿真

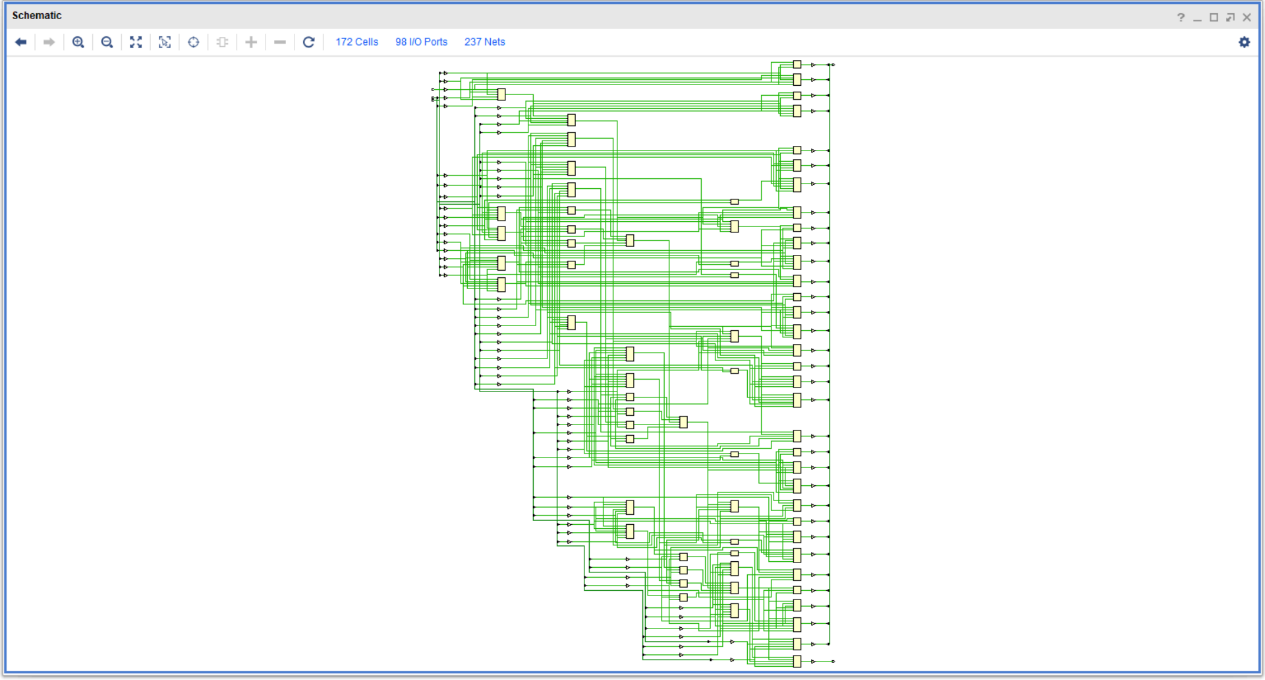


经过检验，加和(s)和进位(co)均无误。

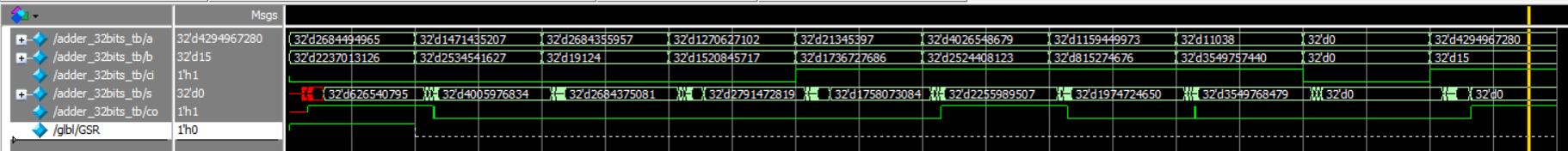
备注：s, b, a都调整为十进制无符号数(unsigned)格式显示，便于核验。co,ci因为只有0,1两个取值所以十六进制也不妨碍观察。

（3）建立vivado工程，对32为加法器进行时序仿真

综合原理图

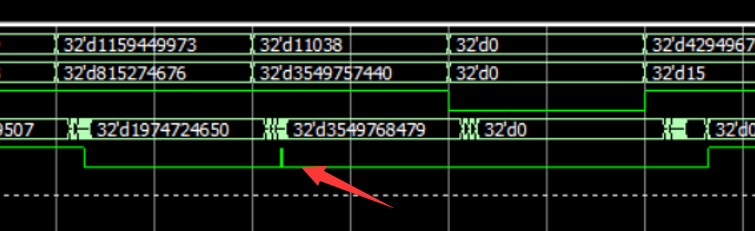


仿真结果图



存在传输延时和建立时间，但**输出稳定后**结果和32位进位选择加法器ModelSim仿真相同，是正确的。

特别需要注意，在计算11038+3549757440时，co有一段为1的脉冲。



在该输入下，因为最后一级无论是否有前级进位都不会有进位的产生，所以认为是竞争冒险现象引起的。

**六、实验中碰到的问题和解决方法**

本次实验比较基础，在过程中并没有遇到太大的问题，只要按照实验书的思路，按步骤完成即可。其中主要的问题是Verilog HDL有些遗忘，导致编译报错，比如赋值语句忘记写assign，忘记模块调用传值的写法等。但是只要根据报错的提示，定位错误，检查语法，参照上学期的代码进行回忆和温习，就可以顺利完成实验。

**七、思考题**

1. 为什么要进行时序仿真？

时序仿真更加贴近电路实际的工作状态。如实验书中提及，32位选择进位加法器存在由进位实验、加和时延、数据选择器时延构成的计算时延。如果出现上一个加法计算的输出尚未稳定，我们就进入了下一拍，开始新一轮两数加法的计算，这样得到的结果是不可靠的。只有进行时序仿真，我们才更有理由判断，我们的稳定输出时间是否足够长，这可以指导我们实际中加法器的使用（即可以作为加法器正常工作时钟频率范围的依据）。另外，我们还能发现电路中是否存在竞争冒险，比如在本实验中，我们就观察到了co为0时出现的1型冒险。

2. 采用流水线技术有什么优缺点？

优点：将电路切分为多个模块，时钟频率只要满足多个模块之间最慢的模块的所需要的时间即可，而不是串行的延迟时间的叠加，从而可以提高时钟频率，进而减少执行时间。同时也增加了数据的吞吐率和硬件的利用率。

缺点：需要考虑数据冒险、结构冒险、控制冒险等依赖问题，电路设计的难度和复杂度更高，而且生产成本、功耗更高。