浙江大学

本科实验报告

课程名称：	计算机组成与设计
姓　　名：	黄嘉欣
学　　院：	信息与电子工程学院
系：	信息与电子工程学系
专　　业：	信息工程
学　　号：	3190102060
指导教师：	屈民军、唐奕

2021 年 10 月 1 日

专业：信息工程
姓名：黄嘉欣
学号：3190102060
日期：2021 年 10 月 1 日
地点：教 11-400

浙江大学实验报告

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：设计性实验

一、实验目的

① 掌握快速加法器的设计方法；

② 熟悉流水线技术；

③ 掌握时序仿真的工作流程。

二、实验任务与要求

① 采用“进位选择加法”技术设计 32 位加法器，并对设计进行功能仿真和时序仿真；

② 采用四级流水线技术设计 32 位加法器，并对设计进行工功能仿真和时序仿真。

三、实验原理

① 四位先行进位加法器的设计：

两个加数分别为$A_3A_2A_1A_0$和$B_3B_2B_1B_0$，C_{-1}为最低位进位。设两个辅助变量分别为$G_3G_2G_1G_0$和$P_3P_2P_1P_0$：$G_i = A_i\&B_i$、$P_i = A_i + B_i$。

一位全加器的逻辑表达式可转化为：

$$\begin{cases} S_i = P_i\overline{G_i} \oplus C_{i-1} \\ C_i = G_i + P_iC_{i-1} \end{cases}$$

利用上述关系，一个四位加法器的进位计算就转化为：

$$\begin{cases} C_0 = G_0 + P_0C_{-1} \\ C_1 = G_1 + P_1C_0 = G_1 + P_1G_0 + P_1P_0C_{-1} \\ C_2 = G_2 + P_2C_1 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_{-1} \\ C_3 = G_3 + P_3C_2 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_{-1} \end{cases}$$

由此可以看出，每一个进位的计算都直接依赖于整个加法器的最初输入，而不需要等待

相邻低位的进位传递。理论上，每一个进位的计算都只需要三个门延迟时间，即产生G_i、P_i的与门和或门，输入为G_i、P_i、C_{-1}的与门，以及最终的或门。同样道理，理论上最终结果 sum 的得到只需要四个门延迟时间。

② 进位选择加法器结构

借鉴并行计算的思想，人们提出了进位选择加法器结构，或者称为有条件的加法器结构，其算法的实质是增加硬件面积换取速度性能的提高。二进制加法的特点是进位或者为逻辑 1，或者为逻辑 0，二者必居其一。将进位链较长的加法器分为 M 块，分别进行加法计算，对除去最低位计算块外的 M-1 块加法结构复制两份，其进位输入分别预定为逻辑 1 和逻辑 0. 于是，M 块加法器可以同时并行进行各自的加法运算，然后根据各自相邻地位加法运算结果产生的进位输出，选择正确的加法结果输出。如图所示为 12 位进位选择加法器的逻辑结构图：

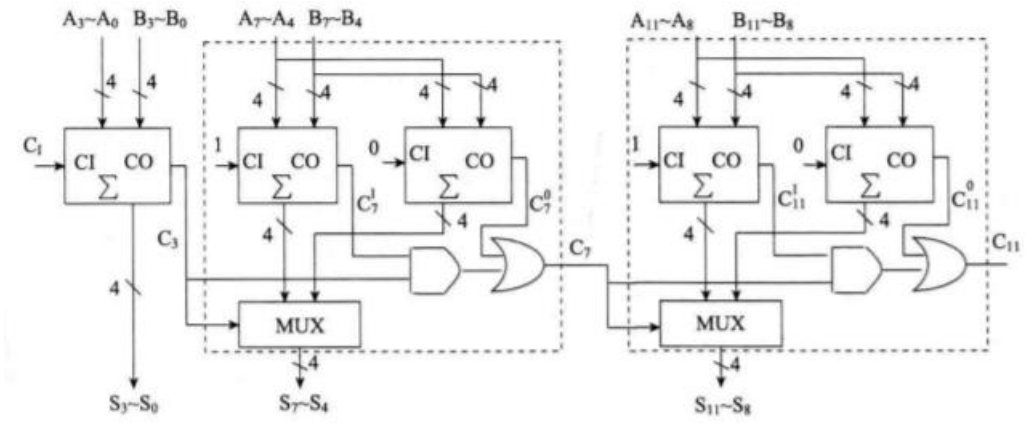

图 3.2 12 位进位选择加法器原理图

③ 流水线加法的设计

在数字系统中，如果完成某些复杂逻辑功能需要较长的时延，就会使系统很难运行在较高的工作频率上。流水线设计是经常用于提高系统运行速度的一种有效方法，即在长时延的逻辑功能块中插入触发器，将复杂的逻辑操作分步进行，减少每个部分的操作，从而提高系统的运行速度。

如图所示为采用 4 级流水线技术的 32 位加法器的原理框图，其采用了 5 级锁存、4 级 8 位加法，整个加法只受 8 位加法器工作速度的限制，从而提高了整个加法器的工作速度。

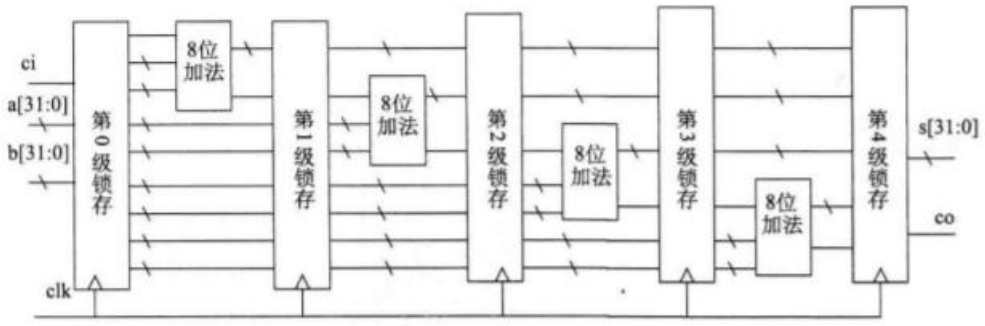

图 3.3 采用 4 级流水线技术的 32 位加法器的原理框图

四、主要仪器设备

装有 Vivado 和 ModelSim SE 软件的计算机。

五、实验内容

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；

② 32 位进位选择加法器的设计：

1) 编写用 32 位进位选择加法器的 Verilog HDL 代码，并用 ModelSim 软件进行功能仿真；

2) 对 32 位加法器进行时序仿真。其步骤如下：

a) 建立 Vivado 工程，除了加入设计文件，还需要加入仿真测试文件，同时设置其属性为 Simulation Only；

b) 对 32 位进位选择加法器进行综合、实现，注意本例不需要约束；

c) 在工程界面的 Flow Navigator 窗口中，右击 SIMULATION，在弹出的快捷菜单中选择 Simulation Setting…命令，将仿真器选择为 ModelSim Simulator，同时加入仿真测试文件，指定仿真库存储路径，单击 OK 按钮；

d) 在工程界面的 FLOW Navigator 窗口中，单击 SIMULATION->Run Simulation 选项，在随后弹出的快捷菜单中选择 Run Post-Implementation Timing Simulation 命令，即可启动时序仿真。

③ 编写采用 4 级流水线技术的 32 位加法器的 Verilog HDL 代码，并对设计进行时序仿真。

六、实验结果与仿真分析

① 4 位先行进位加法器设计

1) 设计代码：

```
module adder_4bits(a, b, ci, s, co);
    input [3:0] a, b; // 输入的四位二进制数
    input ci; // 最低位进位
    output [3:0] s; // 计算的和
    output co; // 进位输出
    wire [2:0] c; // 各级进位
    wire [3:0] p, g; // 辅助变量

    // 计算辅助变量
    assign g = a & b;
    assign p = a | b;

    // 计算各级进位
    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]);

    // 计算求和
    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) 仿真结果与分析：

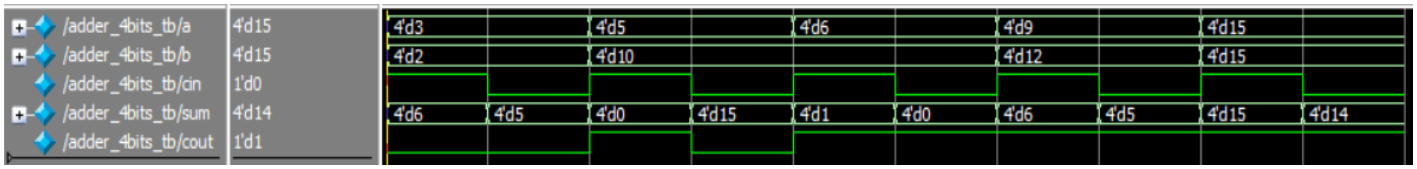

图 6.1 4 位先行进位加法器功能仿真

分析：如图，四位先行加法器计算结果正确，符合预期功能。

② 32 位进位选择加法器设计

1) Verilog HDL 设计：

将 32 位加法器划分为 8 块，最低一块由 4 位超前进位加法器直接构成，后 7 块分别假设前一块的进位为 0 或 1，采用超前进位加法器结构，将两种结果都计算出来，再根据前级进位选择正确的和与进位。此模块由顶层模块 adder_32bits，4 位超前进位加法器 adder_4bits 及二选一数据选择器 mux 组成。

2) 顶层模块设计代码：

```
module adder_32bits(a, b, ci, s, co);
    input [31:0] a, b; // 输入的 32 位数
    input ci; // 输入进位
    output [31:0] s; // 输出的和
    output co; // 输出进位
    wire c0, c1, c2, c3, c4, c5, c6; // 除最后一个计算块外的进位

    // 最低位计算块
    adder_4bits a0(.a(a[3:0]), .b(b[3:0]), .ci(ci), .s(s[3:0]), .co(c0));

    // 后续 7 个计算块
    adder_4bits_block a1(.a(a[7:4]), .b(b[7:4]), .ci(c0), .s(s[7:4]), .co(c1));
    adder_4bits_block a2(.a(a[11:8]), .b(b[11:8]), .ci(c1), .s(s[11:8]), .co(c2));
    adder_4bits_block a3(.a(a[15:12]), .b(b[15:12]), .ci(c2), .s(s[15:12]), .co(c3));
    adder_4bits_block a4(.a(a[19:16]), .b(b[19:16]), .ci(c3), .s(s[19:16]), .co(c4));
    adder_4bits_block a5(.a(a[23:20]), .b(b[23:20]), .ci(c4), .s(s[23:20]), .co(c5));
    adder_4bits_block a6(.a(a[27:24]), .b(b[27:24]), .ci(c5), .s(s[27:24]), .co(c6));
    adder_4bits_block a7(.a(a[31:28]), .b(b[31:28]), .ci(c6), .s(s[31:28]), .co(co));
endmodule
```

3) 功能仿真结果与分析：

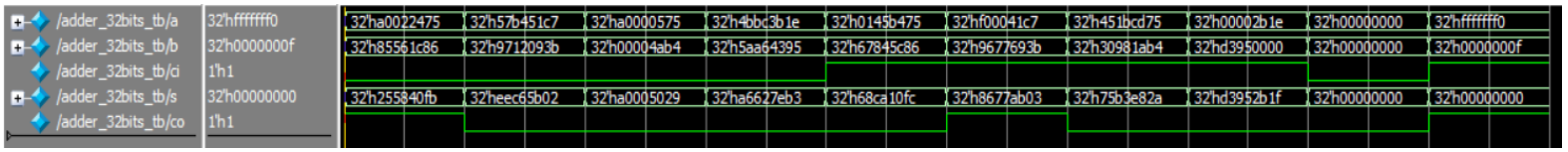

图 6.2.1 32 位进位选择加法器功能仿真（十六进制）

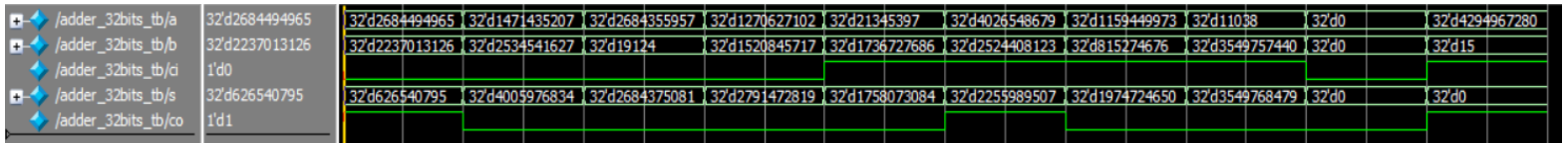

图 6.2.2 32 位进位选择加法器功能仿真（十进制）

分析：如图，将仿真结果与实际计算结果比较可知，加法器计算正确，符合预期功能

4) 时序仿真结果与分析：

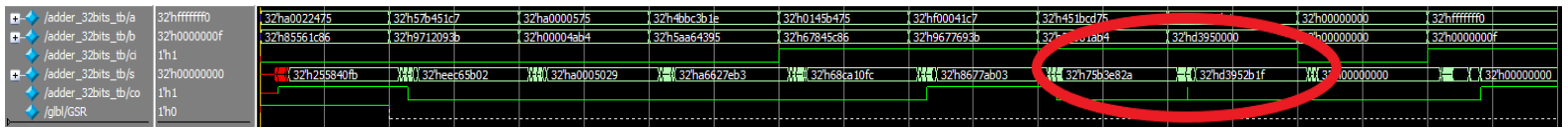

图 6.2.3 32 位进位选择加法器时序仿真

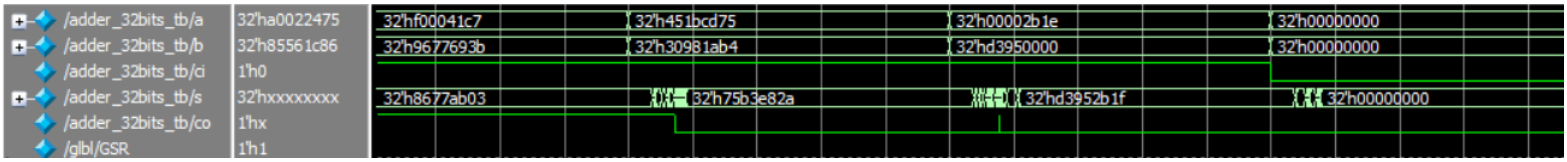

图 6.2.4 32 位进位选择加法器时序仿真（局部放大）

分析：如图，通过局部放大图，可以发现时序仿真输出结果存在比较明显的时延和竞争冒险现象，输出结果变化比输入变化略迟。当输入为 *a*=32'h00002b1e，*b*=32'hd3950000 时，输出进位应当为 0，但实际仿真结果却出现了尖峰，即为 1 型竞争冒险。

③ 采用 4 级流水线技术的 32 位加法器设计

1) Verilog HDL 设计：

将 32 位加法器分为 5 级锁存、4 级 8 位加法，第 0 级锁存初始进位及两个 32 位加数，第 1 级锁存低 8 位求和结果及两个加数的高 24 位，第 2 级锁存低 16 位求和结果及两个加数的高 16 位，以此类推，通过四个时钟周期，即可求得总的 32 位和值及输出进位。此模块由 pipeline_adder 构成。

2) 设计代码：

```
module pipeline_adder(clk, a, b, ci, s, co);
    input clk; // 时钟
    input [31:0] a, b; // 输入的 32 位数
    input ci; // 输入进位
    output reg [31:0] s; // 输出结果（第 4 级锁存）
    output reg co; // 输出进位

    reg [31:0] tmpa0, tmpb0; // 第 0 级锁存 a、b
    reg tmpci; // 第 0 级锁存 ci

    reg [7:0] sum0; // 第 1 级加法结果
    reg tmpc0; // 第 1 级加法进位
    reg [23:0] tmpa1, tmpb1; // 第 1 级锁存 a、b 的前 24 位

    reg [15:0] sum1; // 前 2 级加法结果
    reg tmpc1; // 第 2 级加法进位
    reg [15:0] tmpa2, tmpb2; // 第 2 级锁存 a、b 的前 16 位

    reg [23:0] sum2; // 前 3 级加法结果
    reg tmpc2; // 第 3 级加法进位
    reg [7:0] tmpa3, tmpb3; // 第 3 级锁存 a、b 的前 8 位
```

```
    always @ (posedge clk)
    begin
        tmpa0 <= a;
        tmpb0 <= b;
        tmpci <= ci;   // 锁存 a、b、ci
    end

    // 第 1 级锁存
    always @ (posedge clk)
    begin
        {tmpc0, sum0} <= 9'b0 + tmpa0[7:0] + tmpb0[7:0] + tmpci; // 计算低 8 位和及进位
        tmpa1 <= tmpa0[31:8];
        tmpb1 <= tmpb0[31:8]; // 锁存 a、b 的前 24 位
    end

    // 第 2 级锁存
    always @ (posedge clk)
    begin
        {tmpc1, sum1} <= {9'b0 + tmpa1[7:0] + tmpb1[7:0] + tmpc0, sum0};
        // 低 16 位的和及其进位
        tmpa2 <= tmpa1[23:8];
        tmpb2 <= tmpb1[23:8]; // 锁存 a、b 的前 16 位
    end

    // 第 3 级锁存
    always @ (posedge clk)
    begin
        {tmpc2, sum2} <= {9'b0 + tmpa2[7:0] + tmpb2[7:0] + tmpc1, sum1};
        // 低 24 位的和及其进位
        tmpa3 <= tmpa2[15:8];
        tmpb3 <= tmpb2[15:8]; // 锁存 a、b 的前 8 位
    end

    // 第 4 级锁存
    always @ (posedge clk)
    begin
        {co, s} <= {9'b0 + tmpa3[7:0] +tmpb3[7:0] + tmpc2, sum2}; // 最终求和结果
    end
endmodule
```

3) 功能仿真结果与分析：

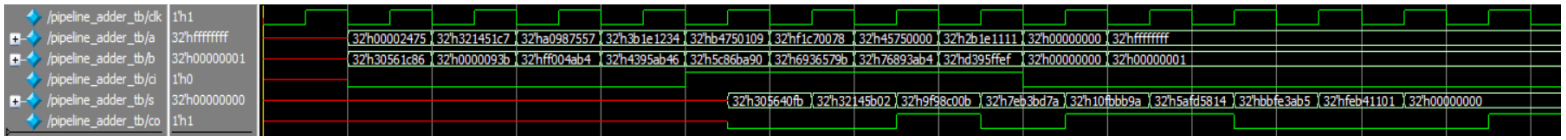

图 6.3.1 采用 4 级流水线技术的 32 位加法器功能仿真

分析：如图，在前四个时钟周期内，输出结果为不定态，这是由于在前四个周期，输入的第一组加数尚未运行完整个流水线，即流水线尚未“填满”。从第五个时钟周期开始，输出结果连续改变，与设计理念相符。通过比较仿真结果与实际运算结果，可以发现加法器计算结果正确，故符合预期功能。

4) 时序仿真结果与分析：

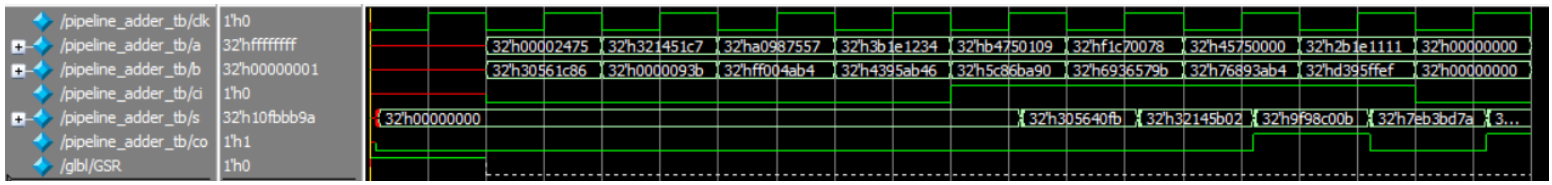

图 6.3.2 采用 4 级流水线技术的 32 位加法器时序仿真

分析：如图，加法器的输出结果存在较为明显的时延现象，且在前四个时钟周期内的输出始终为 0。与选择进位加法器的时序仿真结果比较可以发现，采用流水线技术后，输出时延有所降低，这对于我们提高长时延系统的运行速度可以有所帮助和启发；但也需要看到的是，流水线技术所带来的数据延迟问题在某些情况下可能会导致系统运行效率的降低。

七、思考题

1、为什么要进行时序仿真？

答：因为功能仿真是理想情况下的仿真，反映的只是系统的逻辑功能是否实现，没有与具体开发板相对应。而时序仿真使用布局布线后器件给出的模块和连线的延时信息，在最坏的情况下对电路的行为做出实际估计，反映的是芯片的实际工作状态，与开发板相对应、加载了时延，从而可以在时延、竞争冒险等方面对电路进行仿真，使仿真结果与电路的实际工作状态更加接近，这与实验中的仿真结果是相一致的。

2、采用流水线技术有什么优缺点？

答： ① 优点：流水线技术可以缩短信号必须通过的通路长度，增加数据吞吐量；将复杂的逻辑操作分步进行，从而可以同时对多个数据元素进行不同段的运算，大大提高运算速度；

② 缺点：系统功耗增加，硬件复杂度增加，控制运算更加复杂；需要有“装入时间”和“排空时间”，数据存在延迟。如采用 4 级流水线技术后，32 位加法器延迟了 4 个时钟周期。

八、心得与体会

此次实验，我们学习了先行进位加法器、进位选择加法器等快速加法器的原理和设计方法，熟悉了流水线技术，并首次接触了 Vivado 的时序仿真功能。

总的来说，通过理解、掌握各种加法器的原理及架构，我在设计、编写代码的过程中并没有遇到太多的困难，但有一个令我印象比较深刻：在对流水线加法器进行时序仿真时，得到的第一个输出结果实际上是第二组加数的和，但对其进行功能仿真得到的结果却是正确的值。通过分析，我发现是测试文件没有留初始化的时钟周期，导致触发器初始值不稳定，第一组加数未被成功加载。修改之后，所得结果符合预期。

除此之外，在对进位选择加法器进行设计时，由于数据选择器的地址与输出不对应，仿真得到的结果始终与预期不符。在对模块进行检查调试后，我发现了这个错误并对其进行了改正，最终结果正确。不得不说，我们在设计代码时不能够想当然，不能以为熟悉就可以粗心大意。在后续进行实验时，我会始终注意这个问题，争取减少不必要的错误，得到理想的结果。