专业：电子科学与技术
姓名：
学号：
日期：2023/10/18
地点：教 11 400

浙江大学实验报告

课程名称：计算机组成与设计实验　指导老师：屈民军、唐奕　成绩：
实验名称：实验 8 快速加法器设计

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

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

$$\begin{cases} S_i = P_i\overline{G_i} \oplus C_{i-1} \\ C_i = G_i + P_iC_{i-1} \end{cases} \tag{5.12}$$

利用上述关系，一个四位加法器的进位计算就变转化为

$$\begin{cases} C_0 = G_0 + P_0C_{-1} \\ C_1 = G_1 + P_1C_0 = G_1 + P_1G_0 + P_1P_0C_{-1} \\ C_2 = G_2 + P_2C_1 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_{-1} \\ C_3 = G_3 + P_3C_2 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_{-1} \end{cases} \tag{5.13}$$

由式（5.13）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。

实际上，当加数位数较大时，输入需要驱动的门数较多，其 VLSI 实现的输出时延增加很多，考虑到互连线的延时情况将会更加糟糕。因此，通常在芯片实现时先设计位数较少的超前进位加法器结构，而后以此为基本结构来构造位数较大的加法器。

2．进位选择加法器结构

实际上，由超前进位加法器级联构成的多位加法器只是提高了进位传递的速度，其计算过程与行波进位加法器同样需要等待进位传递的完成。

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（conditional sum adder），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，然后根据各自相邻低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。图5.6 所示为 12 位进位选择加法器的逻辑结构图。12 位加法器划分为 3 块，最低一块（4 位）由 4 位超前进位加法器直接构成，后两块分别假设前一块的进位为 0 或 1 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。如果每一块加法结构内部都采用速度较快的超前进位加法器结构，那么进位选择加法器的计算时延为

$$t_{CSA} = t_{carry} + (M-2)t_{MUX} + t_{sum} \tag{5.14}$$

其中，t_{sum}、t_{carry} 分别为加法器的和与加法器的进位时延，t_{MUX} 为数据选择器的时延。

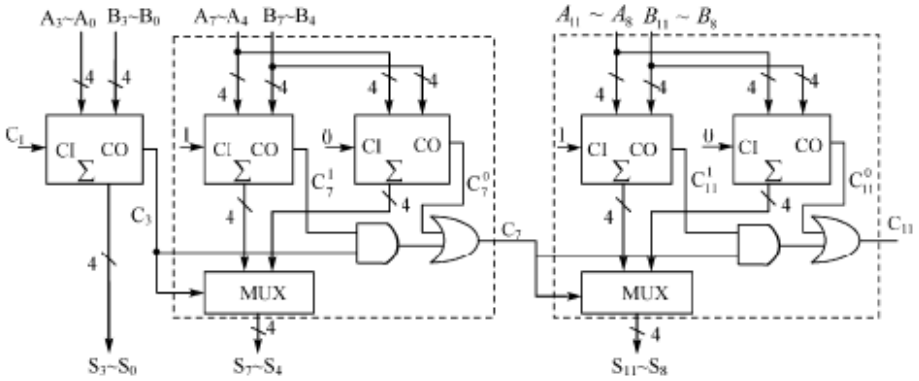


图 5.6　12 位进位选择加法器原理图

四、实验设备

装有 Vivado、ModelSim SE 软件的计算机。

五、实验步骤与过程

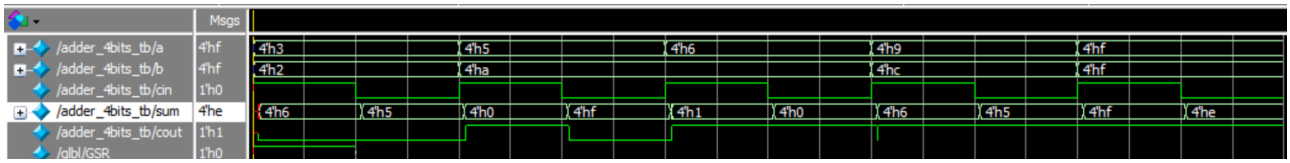
1. 编写 4 位先行进位加法器的 Verilog HDL 代码，并用 ModelSim 软件进行功能仿真。

（1）4 位先行进位加法器的 Verilog HDL 代码

```
`timescale 1ns/10ps
module adder_4bits (
    input [3:0] a,
    input [3:0] b,
    input ci,
    output [3:0] s,
    output co
);
    wire g0, g1, g2, g3;
    wire p0, p1, p2, p3;
    //5.12(2) acquire g[i], p[i]
    assign g0 = a[0]&b[0];
    assign g1 = a[1]&b[1];
    assign g2 = a[2]&b[2];
    assign g3 = a[3]&b[3];
    assign p0 = a[0]|b[0];
    assign p1 = a[1]|b[1];
    assign p2 = a[2]|b[2];
    assign p3 = a[3]|b[3];
```

```
    wire c0, c1, c2, c3;
    //5.13 acquire c[i]
    assign c0 = g0 || (p0&&ci);
    assign c1 = g1 || (p1&&c0);
    assign c2 = g2 || (p2&&c1);
    assign c3 = g3 || (p3&&c2);
    assign co = c3; //output
    //5.12(1) acquire s[i]
    assign s[0] = (p0&&(!g0))^ci;
    assign s[1] = (p1&&(!g1))^c0;
    assign s[2] = (p2&&(!g2))^c1;
    assign s[3] = (p3&&(!g3))^c2;
endmodule
```

（2）ModelSim 仿真



经过检验，加和(sum)和进位(cout)基本无误。在计算 4’h9+4’hc 的时候，cout 输出了约 60ps 的 0，放大发现这段时间的 sum 经历了一系列变化，当 sum=4’ha 时 cout=0，认为是 assign 语句实现的时间略有差异造成的。

2. 32 位进位选择加法器的设计

将 32 位进位选择加法器分成七个子模块，除了第一级为 4 位先行进位加法器，剩下 7 级为两个 4 位先行进位加法器的并联，根据上一级的进位信号输出进位 0,1 的不同结果。

（1）4 位进位选择加法器的 Verilog HDL 代码

```
// conditional sum adder unit
`timescale 1ns/10ps
module csau (
    input [3:0] a,
    input [3:0] b,
    input ci,
    output [3:0] s,
    output co
);
    wire [3:0]s1, s0;
    wire co0, co1;
    adder_4bits a0(.a(a), .b(b), .ci(0), .s(s0), .co(co0));
    adder_4bits a1(.a(a), .b(b), .ci(1), .s(s1), .co(co1));

    mux_4bits mux(.a(s1), .b(s0), .sw(ci), .out(s));
    assign co = co0 || (co1 && ci);
endmodule
```

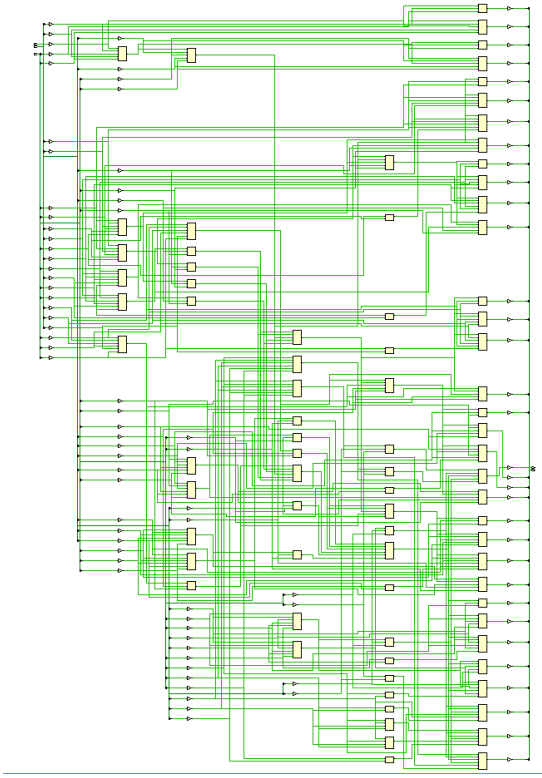
（2）32 位进位选择加法器的 Verilog HDL 代码

```
`timescale 1ns/10ps
module adder_32bits (
    input [31:0] a,
    input [31:0] b,
    input ci,
    output [31:0]s,
    output co
```

```
);
    wire c3, c7, c11, c15, c19, c23, c27, c31;
    adder_4bits a_0to3(.a(a[3:0]), .b(b[3:0]), .ci(ci), .s(s[3:0]), .co(c3));
    csau a_4to7(.a(a[7:4]), .b(b[7:4]), .ci(c3), .s(s[7:4]), .co(c7));
    csau a_8to11(.a(a[11:8]), .b(b[11:8]), .ci(c7), .s(s[11:8]), .co(c11));
    csau a_12to15(.a(a[15:12]), .b(b[15:12]), .ci(c11), .s(s[15:12]), .co(c15));
    csau a_16to19(.a(a[19:16]), .b(b[19:16]), .ci(c15), .s(s[19:16]), .co(c19));
    csau a_20to23(.a(a[23:20]), .b(b[23:20]), .ci(c19), .s(s[23:20]), .co(c23));
    csau a_24to27(.a(a[27:24]), .b(b[27:24]), .ci(c23), .s(s[27:24]), .co(c27));
    csau a_28to31(.a(a[31:28]), .b(b[31:28]), .ci(c27), .s(s[31:28]), .co(co));
endmodule
```

（3）ModelSim 仿真

对工程 run implementation，得到如下原理图：



再 run Post-Implementation Timing Simulation，得到如下时序仿真图：

可见每次运算结果的得出存在一定的延时，但输出稳定后结果均是正确的。

注意到在计算 32h'00002b1e+32'h3950000 时，co 出现了约 0.7ns 的 1 信号，可能跟上述原因一致，也可能是出现了竞争与冒险。

六、实验中碰到的问题和解决方法

本次实验主要的问题是 Verilog HDL 有些遗忘，经过回忆与复习以后便问题不大了。

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。进行时序仿真能够判断结果的稳定输出时间是否足够长，并进一步指导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，时序仿真还能帮助发现电路中是否存在竞争冒险，比如在本实验中，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。