专业：电子科学与技术
姓名：
学号：
日期 2023.9.27
地点：玉泉

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：32 位进位选择加法器设计　实验类型：软件实验　同组学生姓名：无

一、实验目的

1、掌握快速加法器的设计；

2、掌握时序仿真的基本流程。

二、实验任务与要求

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。

三、实验原理与步骤

1、设计基本单元：4 位选择器和 4 位先行加法器

```
module mux4bits(
	input [3:0] inA,
	input [3:0] inB,
	input sel,
	output [3:0] muxOut
	);
	assign muxOut = sel ? inA : inB;
endmodule
```

Code 1　　4 位选择器

设两个加数分别为$A_3A_2A_1A_0$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：

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

利用上述关系，一个 4 位加法器的进位计算就变为：

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

```
module Adder4bits(
	input [3:0] inA,
	input [3:0] inB,
	input cin,
	output reg [3:0] addOut,
	output reg cout
	);
```

装

订

线

```
    wire [3:0] G;
    wire [3:0] P;
    reg [3:0] C;
    assign G = inA & inB;
    assign P = inA | inB;
    always@(inA or inB or cin)
        begin
            C[0] <= cin;
            C[1] <= G[0] | (P[0] & C[0]);
            C[2] <= G[1] | (P[1] & C[1]);
            C[3] <= G[2] | (P[2] & C[2]);
            cout <= G[3] | (P[3] & C[3]);
            addOut <= G^P^C;
        end
endmodule
```

Code 2　　4 位先行进位加法器

每个进位的计算都直接依赖于整个加法器的最初输入，而不需要等待相邻位的进位传递。

2、利用并行计算思想，设计 32 位加法器。

借鉴并行计算的思想，构建进位选择加法器，如图 1。由于二进制加法的进位必为 0 或 1，故将进位链较长的加法器分为多块分别进行加法计算，对除去最低位计算块外的加法结构复制两份，其进位输入分别预定为逻辑 1 和逻辑 0。各部分加法器同时并行各自的加法计算，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。

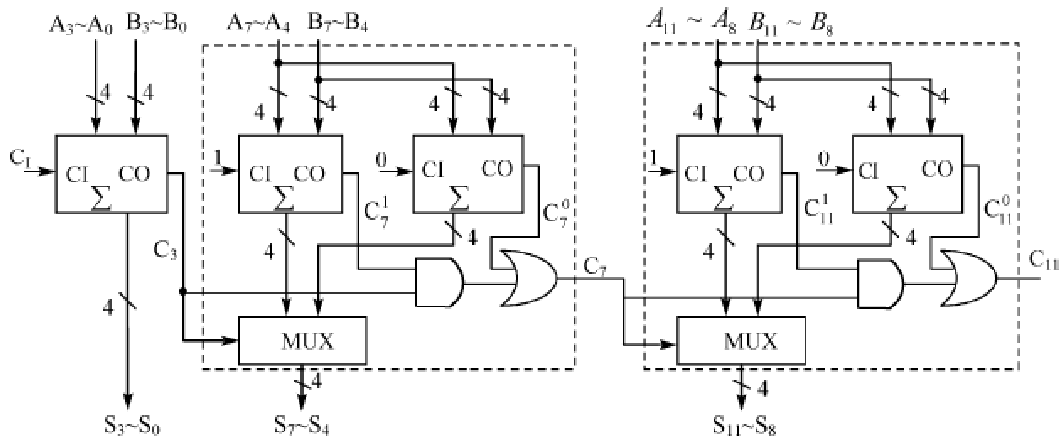


图 1　　进位选择加法器

参照图 1，以 4 位先行进位加法器为基础，设计 32 位加法器。

```
module Adder32bits(
	input [31:0] inA,
	input [31:0] inB,
	input cin,
	output [31:0] addOut,
	output cout
	);
	wire c3,c70,c71,and1out,c7,c111,c110,and2out,c11,c151,c150;
	wire and3out,c15,c191,c190,and4out,c19,c231,c230;
	wire and5out,c23,c271,c270,and6out,c27;
	wire [3:0]adder21, adder30, adder41, adder50,adder61,adder70,adder81;
	wire [3:0]adder90,adder101,adder110,adder121,adder130,adder141,adder150;
	wire c311,c310,and7out;
	/*bits 3:0*/
	Adder4bits Adder1 (.inA(inA[3:0]),.inB(inB[3:0]),.cin(cin),.addOut(addOut[3:0]),.cout(c3));

	/*bits 7:4*/
	Adder4bits Adder2 (.inA(inA[7:4]),.inB(inB[7:4]),.cin(1'b1),.addOut(adder21),.cout(c71));
	Adder4bits Adder3 (.inA(inA[7:4]),.inB(inB[7:4]),.cin(1'b0),.addOut(adder30),.cout(c70));
	mux4bits Mux1(.inA(adder21),.inB(adder30),.sel(c3),.muxOut(addOut[7:4]));
	and and1(and1out,c3,c71);
	or or1(c7,and1out,c70);

	/*bits 11:8*/
	Adder4bits Adder4 (.inA(inA[11:8]),.inB(inB[11:8]),.cin(1'b1),.addOut(adder41),.cout(c111));
	Adder4bits Adder5 (.inA(inA[11:8]),.inB(inB[11:8]),.cin(1'b0),.addOut(adder50),.cout(c110));
	mux4bits Mux2(.inA(adder41),.inB(adder50),.sel(c7),.muxOut(addOut[11:8]));
	and and2(and2out,c7,c111);
	or or2(c11,and2out,c110);

	/*bits 15:12*/
	Adder4bits Adder6 (.inA(inA[15:12]),.inB(inB[15:12]),.cin(1'b1),.addOut(adder61),.cout(c151));
	Adder4bits Adder7 (.inA(inA[15:12]),.inB(inB[15:12]),.cin(1'b0),.addOut(adder70),.cout(c150));
	mux4bits Mux3(.inA(adder61),.inB(adder70),.sel(c11),.muxOut(addOut[15:12]));
	and and3(and3out,c11,c151);
	or or3(c15,and3out,c150);

	/*bits 19:16*/
	Adder4bits Adder8 (.inA(inA[19:16]),.inB(inB[19:16]),.cin(1'b1),.addOut(adder81),.cout(c191));
	Adder4bits Adder9 (.inA(inA[19:16]),.inB(inB[19:16]),.cin(1'b0),.addOut(adder90),.cout(c190));
	mux4bits Mux4(.inA(adder81),.inB(adder90),.sel(c15),.muxOut(addOut[19:16]));
	and and4(and4out,c15,c191);
	or or4(c19,and4out,c190);
```

```
    /*bits 23:20*/
    Adder4bits Adder10 (.inA(inA[23:20]),.inB(inB[23:20]),.cin(1'b1),.addOut(adder101),.cout(c231));
    Adder4bits Adder11 (.inA(inA[23:20]),.inB(inB[23:20]),.cin(1'b0),.addOut(adder110),.cout(c230));
    mux4bits Mux5(.inA(adder101),.inB(adder110),.sel(c19),.muxOut(addOut[23:20]));
    and and5(and5out,c19,c231);
    or or5(c23,and5out,c230);

    /*bits 27:24*/
    Adder4bits Adder12 (.inA(inA[27:24]),.inB(inB[27:24]),.cin(1'b1),.addOut(adder121),.cout(c271));
    Adder4bits Adder13 (.inA(inA[27:24]),.inB(inB[27:24]),.cin(1'b0),.addOut(adder130),.cout(c270));
    mux4bits Mux6(.inA(adder121),.inB(adder130),.sel(c23),.muxOut(addOut[27:24]));
    and and6(and6out,c23,c271);
    or or6(c27,and6out,c270);

    /*bits 31:28*/
    Adder4bits Adder14 (.inA(inA[31:28]),.inB(inB[31:28]),.cin(1'b1),.addOut(adder141),.cout(c311));
    Adder4bits Adder15 (.inA(inA[31:28]),.inB(inB[31:28]),.cin(1'b0),.addOut(adder150),.cout(c310));
    mux4bits Mux7(.inA(adder141),.inB(adder150),.sel(c27),.muxOut(addOut[31:28]));
    and and7(and7out,c27,c311);
    or or7(cout,and7out,c310);
endmodule
```

Code 3　　32 位加法器

四、仿真与测试

选取如下数据进行仿真：

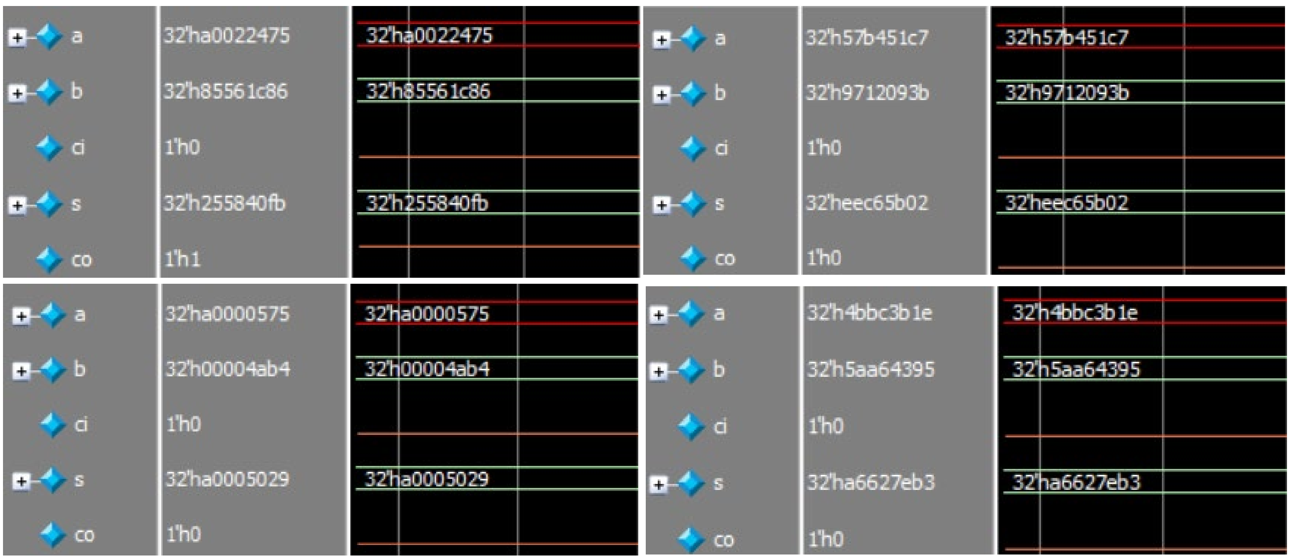
加数 A	加数 B	进位 C_i	理论结果 S	理论溢出位 C_o
32'ha0022475	32'h85561c86	0	32’h255840fb	1
32'h57b451c7	32'h9712093b	0	32’heec65b02	0
32'ha0000575	32'h00004ab4	0	32’ha0005029	0
32'h4bbc3b1e	32'h5aa64395	0	32’ha6627eb3	0
32'h0145b475	32'h67845c86	1	32’b68ca10fc	0
32'hf00041c7	32'h9677693b	1	32’b8677ab02	1
32'h451bcd75	32'h30981ab4	1	32’b75b3e82a	0
32'h00002b1e	b=32'hd3950000	1	32’bd3952b1f	0
0	0	0	0	0
32'hfffffff0	b=32'hf	1	0	1

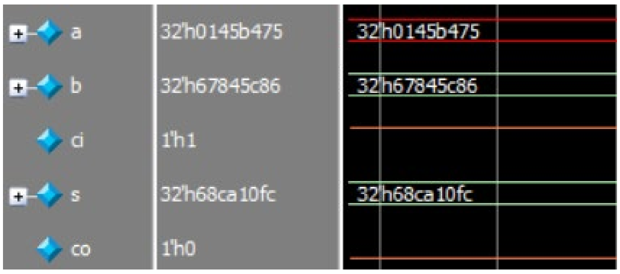
仿真代码如下：

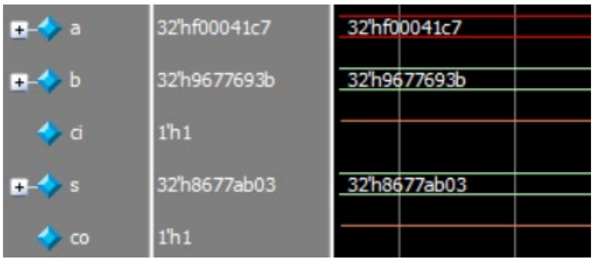
```
module adder_32bits_tb;
reg [31:0] a,b;
reg ci;
wire [31:0]s;
wire co;
parameter DELY=100;
Adder32bits adder_32bits_inst(.inA(a),.inB(b),.cin(ci),.addOut(s),.cout(co));

initial begin
            a=32'ha0022475; b=32'h85561c86;ci=0;
#DELY    a=32'h57b451c7; b=32'h9712093b;ci=0;
#DELY    a=32'ha0000575; b=32'h00004ab4;ci=0;
#DELY    a=32'h4bbc3b1e; b=32'h5aa64395;ci=0;
#DELY    a=32'h0145b475; b=32'h67845c86;ci=1;
#DELY    a=32'hf00041c7; b=32'h9677693b;ci=1;
#DELY    a=32'h451bcd75; b=32'h30981ab4;ci=1;
#DELY    a=32'h00002b1e; b=32'hd3950000;ci=1;
#DELY    a=32'h0;              b=32'h0;      ci=0;
#DELY    a=32'hfffffff0; b=32'hf;      ci=1;
#DELY $stop;
end
endmodule
```

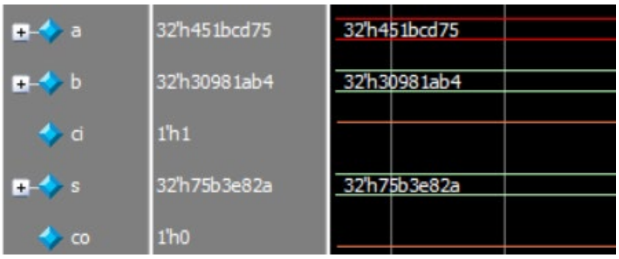
Code 4　　仿真代码

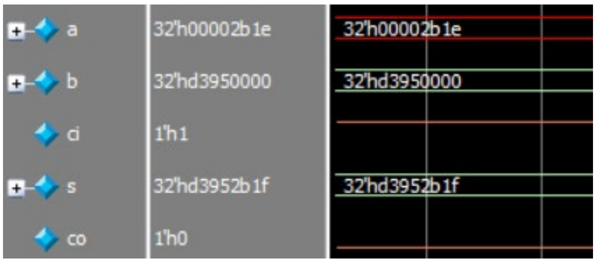
仿真结果如下：

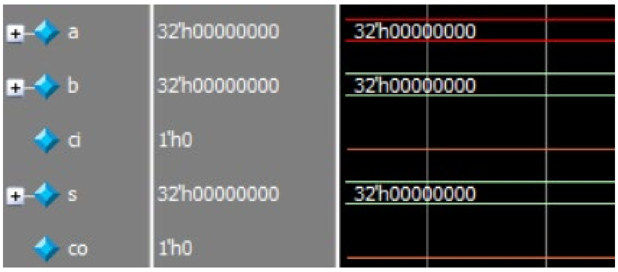


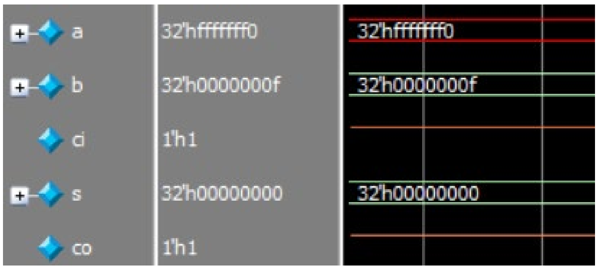












从仿真结果可知，该加法器能够正确进行计算。

五、总结

本实验利用先行进位加法器和并行计算思想实现了 32 位加法器的设计，达到了缩短计算时间、提高程序运行效率的目的。