《数字电路》实验报告

实验名称： 比较器与超前进位加法器 指导教师：王珏，范志华

姓名：韩初晓 学号：2023K8009908002 专业：计算机科学与技术 班级：2306

实验日期：2024.10.17 实验地点：教学楼 224 是否调课/补课：否 成绩：

目录

1 **实验目的** 2
1.1 实验环境 2
2 **实验内容** 2
2.1 实验一：实现四位比较器 2
2.2 实验二：实现十六位比较器 3
2.3 实验三：实现四位加法器 4
2.4 实验四：实现三十二位超前进位加法器 5
3 **实验总结** 6
4 **源代码** 6
4.1 实验一：实现四位比较器 6
4.2 实验二：实现十六位比较器 7
4.3 实验三：实现四位加法器 8
4.4 实验四：实现三十二位超前进位加法器 9

第 1 部分　实验目的

1. 熟悉 Verilog 编程与调试方法。
2. 熟悉简单比较器的工作原理。
3. 通过简单模块例化、连线实现复杂的数字电路。

1.1 实验环境

- Vivado 2017.4 开发工具
- FPGA 开发平台（根据手册中的默认设置进行选择）

第 2 部分　实验内容

2.1 实验一：实现四位比较器

2.1.1 原理说明

四位比较器用于比较两个四位二进制数 A 和 B 的大小关系。模块通过输入信号 in_A_G_B、in_A_E_B、in_A_L_B 控制是否进行比较或直接传递输入值作为输出。比较的结果通过输出信号 out_A_G_B、out_A_E_B 和 out_A_L_B 表示，即当 $A > B$ 时 out_A_G_B 为 1；当 $A = B$ 时 out_A_E_B 为 1；当 $A < B$ 时 out_A_L_B 为 1。控制信号用于决定模块行为，具体如下：

- in_A_E_B：根据 A 和 B 的大小关系输出比较结果。
- in_A_G_B 和 in_A_L_B：直接将输入信号传递至输出，不进行比较。

2.1.2 接口定义

- 输入信号：
  - A[3:0]：四位二进制数，用于比较的第一个数。
  - B[3:0]：四位二进制数，用于比较的第二个数。
  - in_A_G_B：输入控制信号，指示是否直接传递 $A > B$ 的结果。
  - in_A_E_B：输入控制信号，指示是否直接传递 $A = B$ 的结果。
  - in_A_L_B：输入控制信号，指示是否直接传递 $A < B$ 的结果。
- 输出信号：
  - out_A_G_B：表示比较结果，当 $A > B$ 时为 1。
  - out_A_E_B：表示比较结果，当 $A = B$ 时为 1。
  - out_A_L_B：表示比较结果，当 $A < B$ 时为 1。

2.1.3 调试过程及结果

在 comparator_4 模块中，通过编写 Testbench 对其进行仿真测试，验证了模块在不同输入条件下的正确性。仿真波形展示了输入信号 A、B 以及控制信号的变化情况，输出 out_A_G_B、out_A_E_B 和 out_A_L_B 依照输入正确反映了比较结果，仿真结果如下：

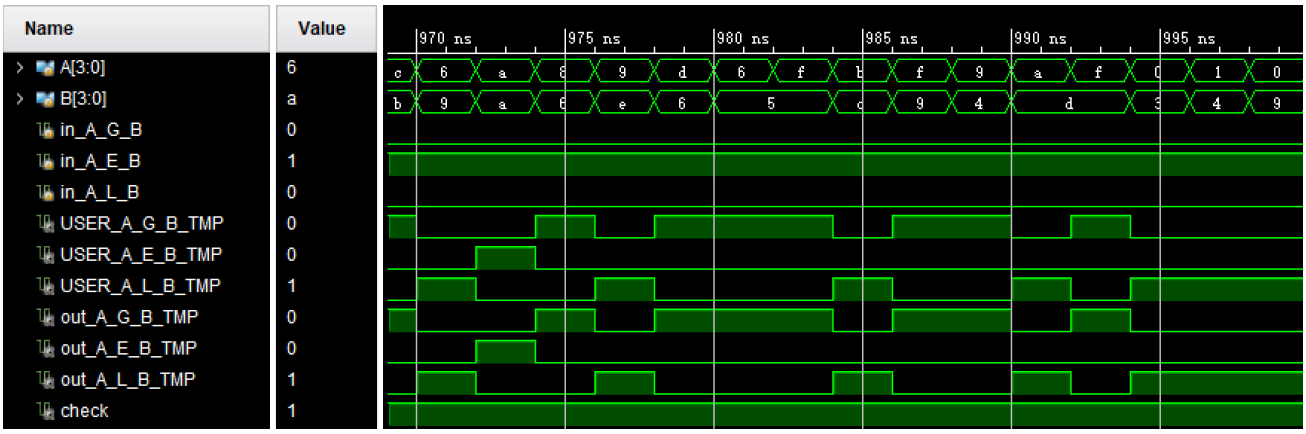

图 1: comparator_4 模块的测试结果

2.2 实验二：实现十六位比较器

2.2.1 原理说明

十六位比较器模块 `comparator_16` 用于比较两个十六位二进制数 A 和 B 的大小关系。该模块将十六位数 A 和 B 分为 4 个 4 位段，并使用四个 `comparator_4` 模块逐级比较每 4 位段的大小关系。通过链式结构实现逐级比较，最终的比较结果输出至 `out_A_G_B`、`out_A_L_B` 和 `out_A_E_B`。模块工作原理如下：

1. 首先比较最高位段 $A[15:12]$ 和 $B[15:12]$ 的大小，将结果传递至下一比较器。2. 接着比较次高位段 $A[11:8]$ 和 $B[11:8]$，基于前一级比较结果继续传递比较结果。3. 同理，比较 $A[7:4]$ 与 $B[7:4]$，以及最低位段 $A[3:0]$ 和 $B[3:0]$。4. 最终，将最低位段的比较结果传递给输出端，完成对十六位二进制数的整体比较。

2.2.2 接口定义

- 输入信号：
  - `A[15:0]`：十六位二进制数，用于比较的第一个数。
  - `B[15:0]`：十六位二进制数，用于比较的第二个数。
  - `in_A_G_B`：输入控制信号，指示是否直接传递 $A > B$ 的结果。
  - `in_A_L_B`：输入控制信号，指示是否直接传递 $A < B$ 的结果。
  - `in_A_E_B`：输入控制信号，指示是否直接传递 $A = B$ 的结果。
- 输出信号：
  - `out_A_G_B`：表示比较结果，当 $A > B$ 时为 1。
  - `out_A_E_B`：表示比较结果，当 $A = B$ 时为 1。
  - `out_A_L_B`：表示比较结果，当 $A < B$ 时为 1。

2.2.3 调试过程及结果

在 `comparator_16` 模块中，通过编写 Testbench 对其进行仿真测试，验证了模块在不同输入条件下的正确性。仿真波形展示了输入信号 A、B 以及中间信号 `result4`、`result3`、`result2` 的变化情况，输出 `out_A_G_B`、`out_A_E_B` 和 `out_A_L_B` 依照输入正确反映了比较结果，仿真结果如下：

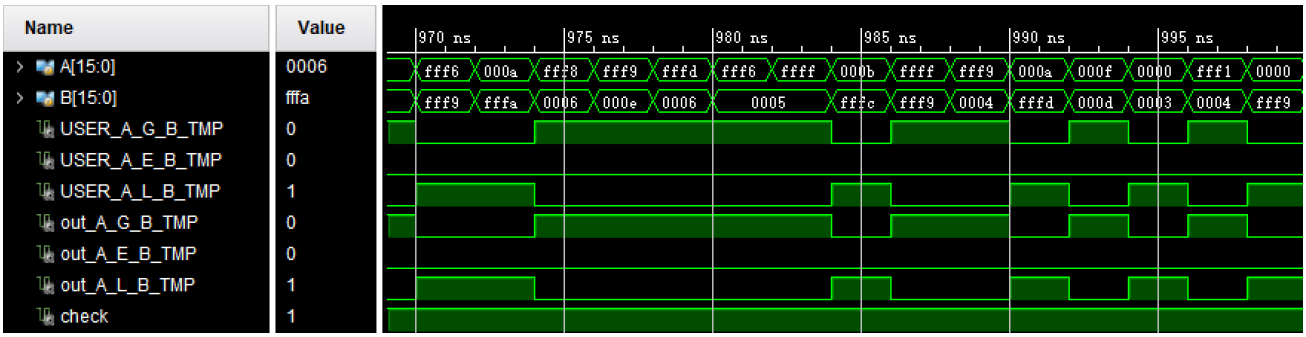

图 2: comparator_16 模块的测试结果

2.3 实验三：实现四位加法器

2.3.1 原理说明

四位加法器模块 `adder_4` 通过输入两个 4 位二进制数 `num_1` 和 `num_2`，以及进位输入 `cin`，计算出 4 位和 `result` 和进位输出 `cout`。模块使用生成信号（g）和传播信号（p）进行加法运算，以优化进位的生成过程。

具体原理如下：1. 计算生成信号 $g[i] = \texttt{num\_1[i]\&num\_2[i]}$ 和传播信号 $p[i] = \texttt{num\_1[i]|num\_2[i]}$。2. 通过组合逻辑生成每一位的进位 $c[i]$，最终得到 `cout`。3. 通过表达式 `result = g & p ^c[3:0]` 得到四位加法结果 `result`。

2.3.2 接口定义

- 输入信号：
  - `cin`：输入进位信号，用于处理来自前一级的进位。
  - `num_1[3:0]`：四位二进制数，第一个加数。
  - `num_2[3:0]`：四位二进制数，第二个加数。
- 输出信号：
  - `result[3:0]`：四位加法运算的结果。
  - `cout`：加法运算的进位输出信号，表示最高位的进位。

2.3.3 调试过程及结果

在 `adder_4` 模块中，通过编写 Testbench 对其进行仿真测试，验证模块在不同输入条件下的正确性。仿真波形展示了输入信号 `num_1`、`num_2`、`cin` 的变化情况，输出 `result` 和 `cout` 正确反映了加法结果和进位输出，仿真结果如下：

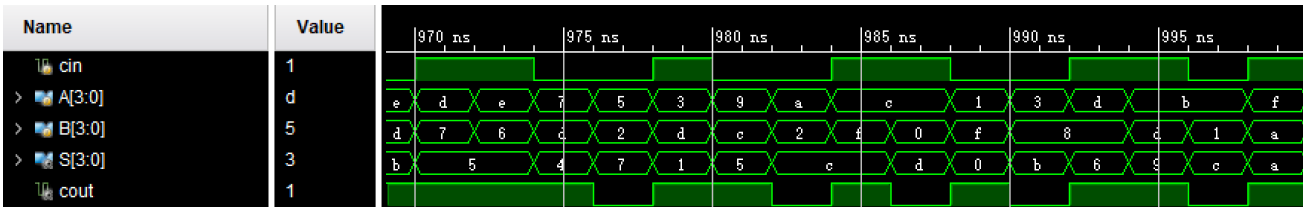

图 3: adder_4 模块的测试结果

2.4 实验四：实现三十二位超前进位加法器

2.4.1 原理说明

三十二位超前进位加法器模块 `add_32` 通过分组加法与进位生成模块实现快速加法运算。该模块由多个子模块构成，包括 4 位加法器、进位生成模块、生成与传播信号计算模块等，以提高计算速度。

具体原理如下：1. 将 32 位输入数 `num_1` 和 `num_2` 分为 8 个 4 位子块。2. 通过连接上层的单元实现（g）和（p）信号的生成，从而快速生成进位。3. 最终输出 32 位加法结果 `result` 和进位输出 `cout`。

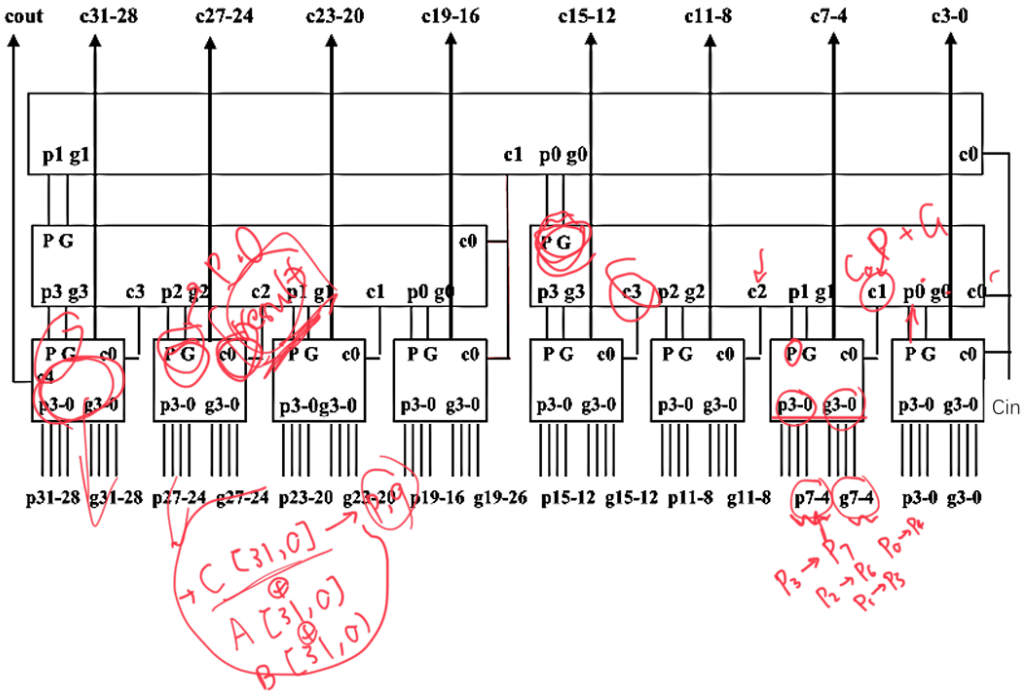

图 4: 32 位超前进位加法器的示意图

2.4.2 接口定义

- 输入信号：
  - `cin`：输入进位信号，用于处理来自前一级的进位。
  - `num_1[31:0]`：32 位二进制数，第一个加数。
  - `num_2[31:0]`：32 位二进制数，第二个加数。
- 输出信号：
  - `result[31:0]`：32 位加法运算的结果。
  - `cout`：加法运算的进位输出信号，表示最高位的进位。

2.4.3 调试过程及结果

在 `add_32` 模块中，通过编写 Testbench 对其进行仿真测试，验证模块在不同输入条件下的正确性。仿真波形展示了输入信号 `num_1`、`num_2`、`cin` 的变化情况，输出 `result` 和 `cout` 正确反映了加法结果和进位输出，仿真结果如下：

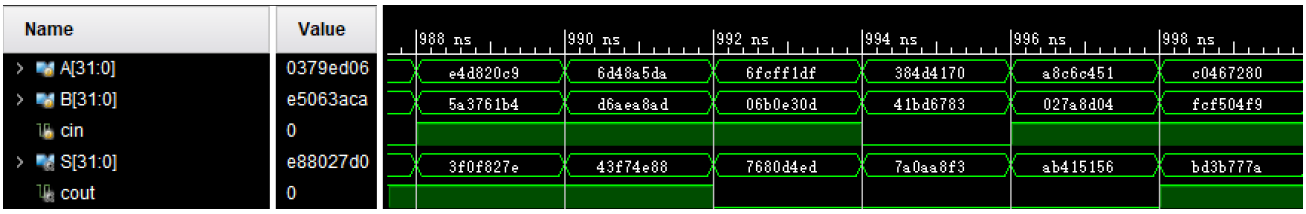

图 5: add_32 模块的测试结果

第 3 部分　实验总结

在本实验中，我通过 Verilog 实现了四个数字电路模块：四位比较器，十六位比较器，四位超前进位加法器和三十二位超前进位加法器。

通过四位和十六位比较器的实现，我更深入地理解了模块在 Verilog 中的实现方法，以及模块之间的连接方式。

在实现四位加法器时，我体会到了代码书写形式不同对实际电路生成结果的影响。例如，表示进位 c 的两个等价的逻辑表达式很可能一个生成出来的电路是串行的，而另一个生成出来的电路则是并行的。如果想要实际生成的电路计算效率高，在书写代码时就应当加以注意。

此外,在实现三十二位超前进位加法器的过程中,我经历了多次试错,最终将四位加法器拆成了 pg_generator, CLU 等模块，并按实际工作的输入输出顺序连接起来，才使得模块正常工作。这启发我在书写代码时应当时刻考虑电路真正的工作过程，否则设计出的电路很可能会产生逻辑错误。

第 4 部分　源代码

4.1 实验一：实现四位比较器

```
1
2  module comparator_4(
3      input [3:0] A,
4      input [3:0] B,
5      output reg out_A_G_B,
6      output reg out_A_E_B,
7      output reg out_A_L_B,
8      input in_A_G_B,
9      input in_A_E_B,
10     input in_A_L_B
11   );
12
13     wire [2:0] control;
14     assign control = {in_A_G_B, in_A_L_B, in_A_E_B};
15
16     always @(*) begin
17       if (control == 3'b001) begin
18         if (A > B) begin
19           out_A_G_B = 1'b1;
20           out_A_L_B = 1'b0;
21           out_A_E_B = 1'b0;
22         end
```

```
        else if (A < B) begin
          out_A_G_B = 1'b0;
          out_A_L_B = 1'b1;
          out_A_E_B = 1'b0;
        end
        else if (A == B) begin
          out_A_G_B = 1'b0;
          out_A_L_B = 1'b0;
          out_A_E_B = 1'b1;
        end
      end
      else if (control == 3'b100) begin
        out_A_G_B = in_A_G_B;
        out_A_L_B = in_A_L_B;
        out_A_E_B = in_A_E_B;
      end
      else if (control == 3'b010) begin
        out_A_G_B = in_A_G_B;
        out_A_L_B = in_A_L_B;
        out_A_E_B = in_A_E_B;
      end
    end
endmodule
```

4.2 实验二：实现十六位比较器

```

module comparator_16(
    input [15:0] A,
    input [15:0] B,
    input in_A_G_B,
    input in_A_L_B,
    input in_A_E_B,
    output out_A_G_B,
    output out_A_L_B,
    output out_A_E_B
    );

    wire [2:0] result4;
    wire [2:0] result3;
    wire [2:0] result2;

    comparator_4 comparator_4_4(
    .A(A[15:12]),
    .B(B[15:12]),
    .in_A_G_B(in_A_G_B),
    .in_A_L_B(in_A_L_B),
    .in_A_E_B(in_A_E_B),
    .out_A_G_B(result4[2]),
    .out_A_L_B(result4[1]),
    .out_A_E_B(result4[0])
```

```
    );

    comparator_4 comparator_4_3(
    .A(A[11:8]),
    .B(B[11:8]),
    .in_A_G_B(result4[2]),
    .in_A_L_B(result4[1]),
    .in_A_E_B(result4[0]),
    .out_A_G_B(result3[2]),
    .out_A_L_B(result3[1]),
    .out_A_E_B(result3[0])
    );

    comparator_4 comparator_4_2(
    .A(A[7:4]),
    .B(B[7:4]),
    .in_A_G_B(result3[2]),
    .in_A_L_B(result3[1]),
    .in_A_E_B(result3[0]),
    .out_A_G_B(result2[2]),
    .out_A_L_B(result2[1]),
    .out_A_E_B(result2[0])
    );

    comparator_4 comparator_4_1(
    .A(A[3:0]),
    .B(B[3:0]),
    .in_A_G_B(result2[2]),
    .in_A_L_B(result2[1]),
    .in_A_E_B(result2[0]),
    .out_A_G_B(out_A_G_B),
    .out_A_L_B(out_A_L_B),
    .out_A_E_B(out_A_E_B)
    );
endmodule
```

4.3 实验三：实现四位加法器

```

module adder_4(
    input cin,
    input [3:0] num_1,
    input [3:0] num_2,
    output [3:0] result,
    output cout
    );

    wire [3:0] p;
    wire [3:0] g;
    wire [4:0] c;

```

```
    assign p = num_1 | num_2;
    assign g = num_1 & num_2;
    assign c[0] = cin;
    assign c[1]=g[0]|cin&p[0];
    assign c[2]=g[1]|g[0]&p[1]|cin&p[1]&p[0];
    assign c[3]=g[2]|g[1]&p[2]|g[0]&p[2]&p[1]|cin&p[2]&p[1]&p[0];
    assign c[4]=g[3]|g[2]&p[3]|g[1]&p[3]&p[2]|g[0]&p[3]&p[2]&p[1]|cin&p[3]&p[2]&p[1]&p[0];

    assign result = ~ g & p ^ c[3:0];
    assign cout = c[4];

endmodule
```

4.4 实验四：实现三十二位超前进位加法器

```

module adder_component_clu(
  input [3:0] g,
  input [3:0] p,
  input ci,
  output [3:0] c
);
    assign c[0]=g[0]|ci&p[0];
    assign c[1]=g[1]|g[0]&p[1]|ci&p[1]&p[0];
    assign c[2]=g[2]|g[1]&p[2]|g[0]&p[2]&p[1]|ci&p[2]&p[1]&p[0];
    assign c[3]=g[3]|g[2]&p[3]|g[1]&p[3]&p[2]|g[0]&p[3]&p[2]&p[1]|ci&p[3]&p[2]&p[1]&p[0];

endmodule

module adder_component_tu(
  input [3:0] g,
  input [3:0] p,
  output [3:0] t
);
  assign t = ~g & p;

endmodule

module adder_component_pg_generator(
  input a,
  input b,

  output g,
  output p
);
  assign g = a & b;
  assign p = a | b;

endmodule

module adder_4m1(
```

```
    input cin,
    input [3:0] num_1,
    input [3:0] num_2,
    output [3:0] result,
    output [3:0] c
    //output pm,
    //output gm
    );

    wire [3:0] p_cla;
    wire [3:0] g_cla;
    wire [3:0] t_cla;
    wire [3:0] co_clu;

    adder_component_pg_generator PG0 (
    .a(num_1[0]),
    .b(num_2[0]),
    .g(g_cla[0]),
    .p(p_cla[0])
    );

    adder_component_pg_generator PG1 (
      .a(num_1[1]),
      .b(num_2[1]),
      .g(g_cla[1]),
      .p(p_cla[1])
    );

    adder_component_pg_generator PG2 (
      .a(num_1[2]),
      .b(num_2[2]),
      .g(g_cla[2]),
      .p(p_cla[2])
    );

    adder_component_pg_generator PG3 (
      .a(num_1[3]),
      .b(num_2[3]),
      .g(g_cla[3]),
      .p(p_cla[3])
    );

    adder_component_tu TU(
      .g(g_cla),
      .p(p_cla),
      .t(t_cla)
    );

    adder_component_clu CLU(
      .p(p_cla),
      .g(g_cla),
      .ci(cin),
```

```
      .c(co_clu)
    );

    assign result[0] = t_cla[0] ^ cin;
    assign result[1] = t_cla[1] ^ co_clu[0];
    assign result[2] = t_cla[2] ^ co_clu[1];
    assign result[3] = t_cla[3] ^ co_clu[2];
    assign c = {co_clu[2], co_clu[1], co_clu[0], cin};
   // assign pm = g_cla[3] | (p_cla[3] & g_cla[2]) | (p_cla[3] & p_cla[2] & g_cla[1]) | (
       p_cla[3] & p_cla[2] & p_cla[1] & g_cla[0]);
   // assign gm = p_cla[3] & p_cla[2] & p_cla[1] & p_cla[0];
endmodule

module pggenerator(
    input cin,
    input [3:0] p,
    input [3:0] g,
    output pm,
    output gm,
    output [3:0] c
    );

    wire [3:0] co_clu;

    adder_component_clu CLU(
      .p(p),
      .g(g),
      .ci(cin),
      .c(co_clu)
    );

    assign c[0] = cin;
    assign c[1] = co_clu[0];
    assign c[2] = co_clu[1];
    assign c[3] = co_clu[2];
    assign pm = g[3] | (p[3] & g[2]) | (p[3] & p[2] & g[1]) | (p[3] & p[2] & p[1] & g[0]);
    assign gm = p[3] & p[2] & p[1] & p[0];
endmodule

module cla_component_pgm_generator(
  input [3:0] num_1,
  input [3:0] num_2,

  output pm,
  output gm
);

    wire [3:0] g;
    wire [3:0] p;

      adder_component_pg_generator PG0 (
    .a(num_1[0]),
```

```
    .b(num_2[0]),
    .g(g[0]),
    .p(p[0])
    );

    adder_component_pg_generator PG1 (
      .a(num_1[1]),
      .b(num_2[1]),
      .g(g[1]),
      .p(p[1])
    );

    adder_component_pg_generator PG2 (
      .a(num_1[2]),
      .b(num_2[2]),
      .g(g[2]),
      .p(p[2])
    );

    adder_component_pg_generator PG3 (
      .a(num_1[3]),
      .b(num_2[3]),
      .g(g[3]),
      .p(p[3])
    );

  assign gm=g[3]|g[2]&p[3]|g[1]&p[3]&p[2]|g[0]&p[3]&p[2]&p[1];
    assign pm=p[3]&p[2]&p[1]&p[0];

endmodule

module cla_generate_from_pg (
  input [3:0] g,
    input [3:0] p,

    output gm,
    output pm
);

    assign gm=g[3]|g[2]&p[3]|g[1]&p[3]&p[2]|g[0]&p[3]&p[2]&p[1];
    assign pm=p[3]&p[2]&p[1]&p[0];

endmodule

module add_32(
      input [31:0] num_1,
      input [31:0] num_2,
      input cin,
      output cout,
      output [31:0] result
    );
      wire [31:0] c;
```

```
      wire [15:0] p_and_g;
      wire [3:0] p_and_g_11;
      wire [3:0] carry_9;
      wire [3:0] carry_10;
      wire [3:0] carry_11;

    adder_4m1 adder_4m_1(
      .cin(cin),
      .num_1(num_1[3:0]),
      .num_2(num_2[3:0]),
      .result(result[3:0]),
      //.pm(p_and_g[1]),
      //.gm(p_and_g[0]),
      .c(c[3:0])
    );

    cla_component_pgm_generator PG1 (
      .num_1(num_1[3:0]),
      .num_2(num_2[3:0]),
      .pm(p_and_g[1]),
      .gm(p_and_g[0])
    );


    adder_4m1 adder_4m_2(
      .cin(carry_9[1]),
      .num_1(num_1[7:4]),
      .num_2(num_2[7:4]),
      .result(result[7:4]),
//      .pm(p_and_g[3]),
//      .gm(p_and_g[2]),
      .c(c[7:4])
    );

    cla_component_pgm_generator PG2 (
      .num_1(num_1[7:4]),
      .num_2(num_2[7:4]),
      .pm(p_and_g[3]),
      .gm(p_and_g[2])
    );

    adder_4m1 adder_4m_3(
      .cin(carry_9[2]),
      .num_1(num_1[11:8]),
      .num_2(num_2[11:8]),
      .result(result[11:8]),
      //.pm(p_and_g[5]),
      //.gm(p_and_g[4]),
      .c(c[11:8])
    );

    cla_component_pgm_generator PG3 (
```

```
244       .num_1(num_1[11:8]),
245       .num_2(num_2[11:8]),
246       .pm(p_and_g[5]),
247       .gm(p_and_g[4])
248     );
249
250     adder_4m1 adder_4m_4(
251       .cin(carry_9[3]),
252       .num_1(num_1[15:12]),
253       .num_2(num_2[15:12]),
254       .result(result[15:12]),
255       //.pm(p_and_g[7]),
256       //.gm(p_and_g[6]),
257       .c(c[15:12])
258     );
259
260     cla_component_pgm_generator PG4 (
261       .num_1(num_1[15:12]),
262       .num_2(num_2[15:12]),
263       .pm(p_and_g[7]),
264       .gm(p_and_g[6])
265     );
266
267     adder_4m1 adder_4m_5(
268       .cin(carry_11[1]),
269       .num_1(num_1[19:16]),
270       .num_2(num_2[19:16]),
271       .result(result[19:16]),
272       //.pm(p_and_g[9]),
273       //.gm(p_and_g[8]),
274       .c(c[19:16])
275     );
276
277    cla_component_pgm_generator PG5 (
278       .num_1(num_1[19:16]),
279       .num_2(num_2[19:16]),
280       .pm(p_and_g[9]),
281       .gm(p_and_g[8])
282     );
283
284     adder_4m1 adder_4m_6(
285       .cin(carry_10[1]),
286       .num_1(num_1[23:20]),
287       .num_2(num_2[23:20]),
288       .result(result[23:20]),
289       //.pm(p_and_g[11]),
290       //.gm(p_and_g[10]),
291       .c(c[23:20])
292     );
293
294     cla_component_pgm_generator PG6 (
295       .num_1(num_1[23:20]),
```

```
296       .num_2(num_2[23:20]),
297       .pm(p_and_g[11]),
298       .gm(p_and_g[10])
299     );
300
301     adder_4m1 adder_4m_7(
302       .cin(carry_10[2]),
303       .num_1(num_1[27:24]),
304       .num_2(num_2[27:24]),
305       .result(result[27:24]),
306       //.pm(p_and_g[13]),
307       //.gm(p_and_g[12]),
308       .c(c[27:24])
309     );
310
311     cla_component_pgm_generator PG7 (
312       .num_1(num_1[27:24]),
313       .num_2(num_2[27:24]),
314       .pm(p_and_g[13]),
315       .gm(p_and_g[12])
316     );
317
318     adder_4m1 adder_4m_8(
319       .cin(carry_10[3]),
320       .num_1(num_1[31:28]),
321       .num_2(num_2[31:28]),
322       .result(result[31:28]),
323       //.pm(p_and_g[15]),
324       //.gm(p_and_g[14]),
325       .c(c[31:28])
326     );
327
328     cla_component_pgm_generator PG8 (
329       .num_1(num_1[31:28]),
330       .num_2(num_2[31:28]),
331       .pm(p_and_g[15]),
332       .gm(p_and_g[14])
333     );
334
335     cla_generate_from_pg PG9 (
336       .p({p_and_g[7], p_and_g[5], p_and_g[3], p_and_g[1]}),
337       .g({p_and_g[6], p_and_g[4], p_and_g[2], p_and_g[0]}),
338       .pm(p_and_g_11[1]),
339       .gm(p_and_g_11[0])
340     );
341
342     pggenerator pggenerator_9(
343       .p({p_and_g[7], p_and_g[5], p_and_g[3], p_and_g[1]}),
344       .g({p_and_g[6], p_and_g[4], p_and_g[2], p_and_g[0]}),
345       //.pm(p_and_g_11[1]),
346       //.gm(p_and_g_11[0]),
347       .cin(cin),
```

```
348       .c(carry_9)
349     );
350 
351     pggenerator pggenerator_10(
352       .p({p_and_g[15], p_and_g[13], p_and_g[11], p_and_g[9]}),
353       .g({p_and_g[14], p_and_g[12], p_and_g[10], p_and_g[8]}),
354       .pm(p_and_g_11[3]),
355       .gm(p_and_g_11[2]),
356       .cin(carry_11[1]),
357       .c(carry_10)
358     );
359 
360     pggenerator pggenerator_11(
361       .p({1'b0, 1'b0, 1'b0, p_and_g_11[1]}),
362       .g({1'b0, 1'b0, 1'b0, p_and_g_11[0]}),
363       .pm(),
364       .gm(),
365       .cin(cin),
366       .c(carry_11)
367     );
368 
369     assign cout = c[31];
370 
371 endmodule
```