DD-Lab project 3

verilog

```
`timescale 1ns/100ps

module FA(s,p,g,co,a,b,ci);
    output s,p,g,co;
    input a,b,ci;
    wire w1,w2,w3;

    // lookahead
    or #15 G1(p,a,b);
    and #10 G2(g,a,b);

    // sum
    xor #20 G3(w1,a,b);
    xor #20 G4(s,w1,ci);

    // carry
    and #10 G5(w2,a,ci);
    and #10 G6(w3,b,ci);
    or #20 G7(co,g,w2,w3);
endmodule

module t_FA;
    wire s,p,g,co;
    reg a,b,ci;

    FA addr(s,p,g,co,a,b,ci);

    initial begin
        a = 1'b0; b = 1'b0; ci = 1'b0;
        #100 a = 1'b0; b = 1'b0; ci = 1'b1;
        #100 a = 1'b0; b = 1'b1; ci = 1'b0;
        #100 a = 1'b0; b = 1'b1; ci = 1'b1;
        #100 a = 1'b1; b = 1'b0; ci = 1'b0;
        #100 a = 1'b1; b = 1'b0; ci = 1'b1;
        #100 a = 1'b1; b = 1'b1; ci = 1'b0;
        #100 a = 1'b1; b = 1'b1; ci = 1'b1;
    end

    // initial #1000 $finish;
    initial $dumpvars;

endmodule

module RCA_16 (
    co, s, c0, a, b
);
    output co;
    output [15:0] s;
    input c0;
    input [15:0] a,b;
    wire [16:0] c;

    assign c[0] = c0;
    assign co = c[16];
```

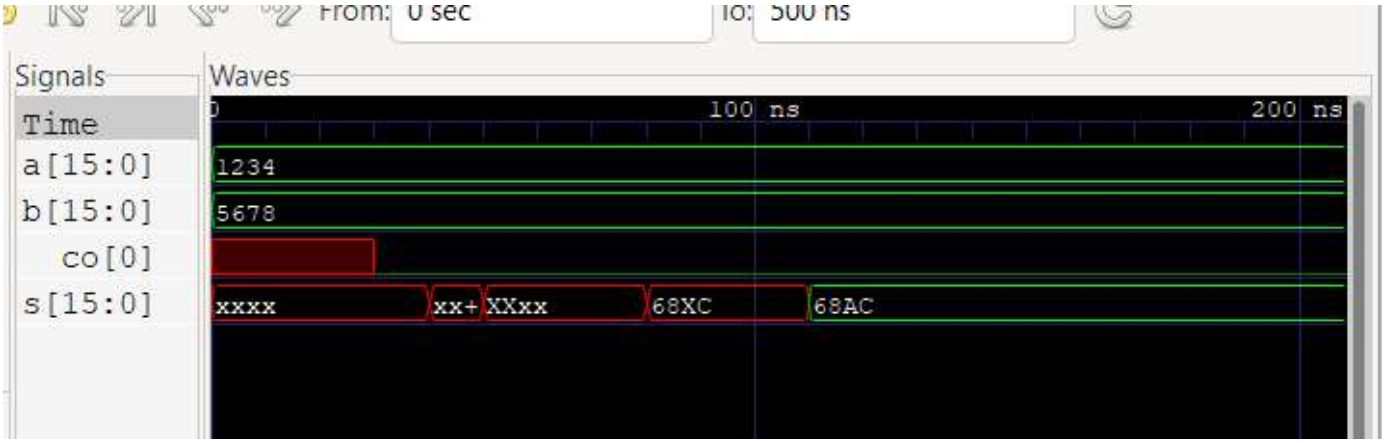
```
    genvar i;
    generate
        for (i = 0; i<16 ; i = i + 1 ) begin
            FA adder (.s(s[i]), .co(c[i+1]), .a(a[i]), .b(b[i]), .ci(c[i]) );
        end
    endgenerate

endmodule

module t_RCA_16;

    wire co;
    wire [15:0] s;
    reg [15:0] a,b;

    RCA_16 rca (co,s,1'b0,a,b);

    initial begin
        a = 16'h1234; b = 16'h5678;
    end
    initial $dumpvars;
    // initial #500 $finish;

endmodule

module CLA_4 (
    co, s, po, go, c0, a, b
);

    output co, po, go;
    output [3:0] s;
    input [3:0] a, b;
    input c0;
    wire [4:0] c,p,g;

    assign co = c[4];
    assign c[0] = c0;

    genvar i;
    generate
        for (i = 0;i<4 ;i=i+1 ) begin
            FA addr(.s(s[i]),.p(p[i]),.g(g[i]),.a(a[i]),.b(b[i]),.ci(c[i]));
        end
    endgenerate

    // LCU
    wire wp0c0,wp1g0,wp1p0c0, wp2g1, wp2p1g0, wp2p1p0c0;

    and #10 p0c0(wp0c0, p[0], c[0]);
    or #15 c1(c[1], g[0], wp0c0);

    and #10 p1g0(wp1g0, p[1], g[0]);
    and #15 p1p0c0(wp1p0c0, p[1], p[0], c[0]);
    or #20 c2(c[2], g[1], wp1p0c0, wp1g0);
```

```
    and #10 p2g1(wp2g1, p[2], g[1]);
    and #15 p2p1g0(wp2p1g0, p[2], p[1], g[0]);
    and #20 p2p1p0c0(wp2p1p0c0, p[2], p[1], p[0], c[0]);
    or #25 c3(c[3], g[2], wp2g1, wp2p1g0, wp2p1p0c0);

    and #10 p3g2(wp3g2, p[3], g[2]);
    and #15 p3p2g1(wp3p2g1, p[3], p[2], g[1]);
    and #20 p3p2p1g0(wp3p2p1g0, p[3], p[2], p[1], g[0]);
    and #25 p3p2p1p0c0(wp3p2p1p0c0, p[3], p[2], p[1], p[0], c[0]);
    or #30 c4(c[4], g[3], wp3g2,wp3p2g1,wp3p2p1g0,wp3p2p1p0c0);

    and #20 Gpo(po, p[3], p[2], p[1], p[0]);

    or #25 Ggo(go, g[3], wp3g2, wp3p2g1, wp3p2p1g0);

endmodule

module t_CLA_4;

    wire [3:0] s;
    wire p, g, c;
    reg [3:0] a,b;

    CLA_4 cla(c, s, p, g, 1'b0, a, b);

    initial begin
        a = 4'h3; b = 4'h7;
        #200 a = 4'h9; b = 4'h9;
        #200 a = 4'h8; b = 4'h7;
    end
    initial $dumpvars;
    // initial #600 $finish;

endmodule

module CLA_16 (
    co, s, po, go, ci, a, b
);
    output co, po, go;
    output [15:0] s;
    input ci;
    input [15:0] a,b;

    wire [3:0] p,g;
    wire [4:0] c;
    assign c[0] = ci;
    assign co = c[4];

    genvar i;
    generate
        for (i = 0;i<4 ;i=i+1 ) begin
            CLA_4 adder(.s(s[4*(i+1)-1:4*i]), .po(p[i]), .go(g[i]), .a(a[4*(i+1)-1:4*i])
        end
    endgenerate
```

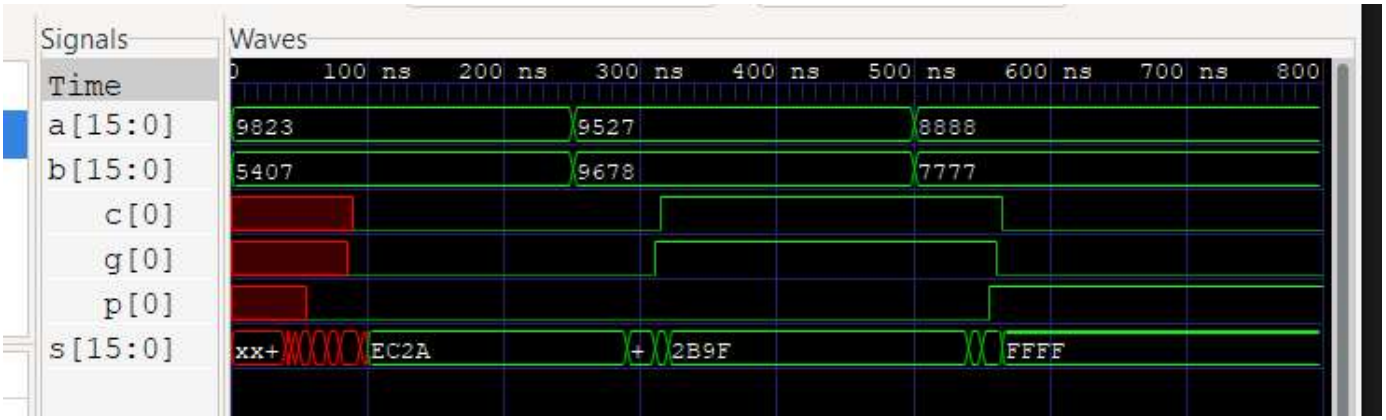
```
    //LCU
    wire wp0c0,wp1g0,wp1p0c0, wp2g1, wp2p1g0, wp2p1p0c0;

    and #10 p0c0(wp0c0, p[0], c[0]);
    or #15 c1(c[1], g[0], wp0c0);

    and #10 p1g0(wp1g0, p[1], g[0]);
    and #15 p1p0c0(wp1p0c0, p[1], p[0], c[0]);
    or #20 c2(c[2], g[1], wp1p0c0, wp1g0);

    and #10 p2g1(wp2g1, p[2], g[1]);
    and #15 p2p1g0(wp2p1g0, p[2], p[1], g[0]);
    and #20 p2p1p0c0(wp2p1p0c0, p[2], p[1], p[0], c[0]);
    or #25 c3(c[3], g[2], wp2g1, wp2p1g0, wp2p1p0c0);

    and #10 p3g2(wp3g2, p[3], g[2]);
    and #15 p3p2g1(wp3p2g1, p[3], p[2], g[1]);
    and #20 p3p2p1g0(wp3p2p1g0, p[3], p[2], p[1], g[0]);
    and #25 p3p2p1p0c0(wp3p2p1p0c0, p[3], p[2], p[1], p[0], c[0]);
    or #30 c4(c[4], g[3], wp3g2,wp3p2g1,wp3p2p1g0,wp3p2p1p0c0);

    and #20 Gpo(po, p[3], p[2], p[1], p[0]);

    or #25 Ggo(go, g[3], wp3g2, wp3p2g1, wp3p2p1g0);

endmodule

module t_CLA_16;

    wire [15:0] s;
    wire p, g, c;
    reg [15:0] a,b;

    CLA_16 cla(c, s, p, g, 1'b0, a, b);

    initial begin
        a = 16'h9823; b = 16'h5407;
        #250 a = 16'h9527; b = 16'h9678;
        #250 a = 16'h8888; b = 16'h7777;
    end
    initial $dumpvars;
    // initial #800 $finish;

endmodule

module compare_CLA_RCA;

    wire c_cla, c_rca;
    wire [15:0] s_cla, s_rca;
    reg [15:0] a, b;

    CLA_16 cla(.co(c_cla), .s(s_cla), .ci(1'b0), .a(a), .b(b));
    RCA_16 rca(.co(c_rca), .s(s_rca), .c0(1'b0), .a(a), .b(b));
```

```
        initial begin
            a = 16'h6677; b = 16'h2233;
            #3000 a = 16'hABCD; b = 16'hF39C;
            #3000 a = 16'h8888; b = 16'h7777;
        end
        initial $dumpvars;
        initial #9000 $finish;

    endmodule
```

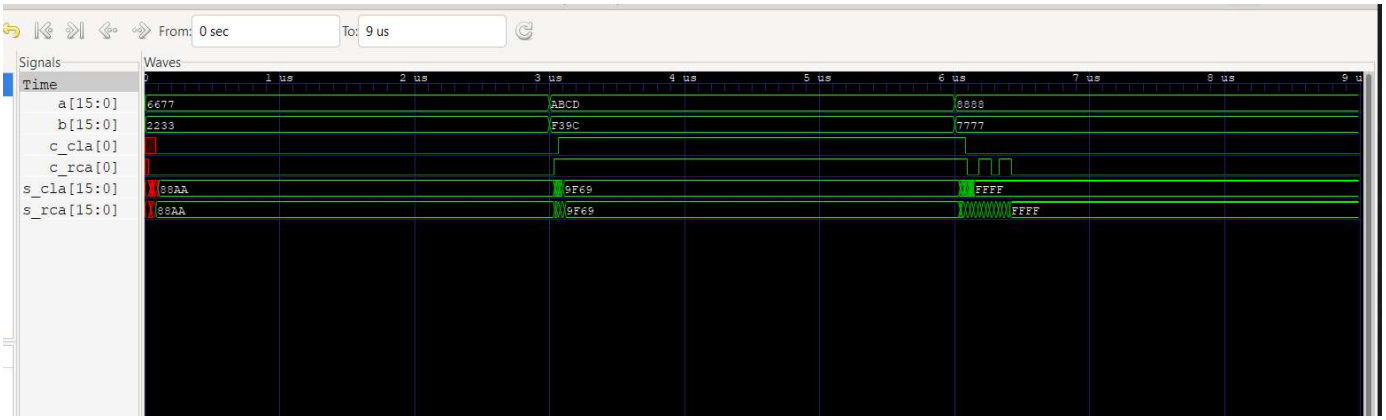
RCA test result



CLA test result



CLA vs. RCA



Is CLA always faster than RCA?

- 否

If not, explain your simulation results.

- 實驗數據中 RCA 出現結果比 CLA 快的原因是該組測資需要進位的位元不多，因此很快出現正確的結果，然而這並不代表當時的值就是可信的，因為有可能較低位元進位造成高位元輸出改變，影響最後結果。

> If the circuit delay is not a fixed value, which one should we care about?

- 經由計算及完整測試後得出的結果，即保證輸出是正確值所需的時間

> Judge the two adders in terms of cost and performance.

- 處理 n bit 時，RCA 需要等待十個 1 bit FA 依序運算完成才能確定結果，複雜度為 O(n) (FA 複雜度為 O(1))；而 CLA 複雜度為 LCU(n) + FA(1) = O(LCU(n))，而 LCU 視情況使用一層 SOP (O(1)) 或使用多個 4 bit LCU (\(O(\log_4 n)\)) 組合，所以複雜度為 O(log n)。由此可知，平均來說 CLA 會比 RCA 更快，尤其是位元數 (n) 越大時。然而，比起 RCA，CLA 需要更多電路。RCA 在處理 n bit 運算時只需要 n 個 FA，複雜度為 O(n)；CLA 則還需要 LCU，LCU 的複雜度是 \(O(n^2)\)，即使使用 4 bit LCU 組合，元件複雜度也要 \(O(n \log n)\)