```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
////////
// Company:
// Engineer:
//
// Create Date: 01.07.2025 15:02:47
// Design Name:
// Module Name: rcomplement
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
////////


module rcomplement (
    input  [3:0] a,
    input        cin1, cin2, r,
    output [3:0] y
);
    // Inverted bits (1's complement)
    wire w0, w1, w2, w3;
    not (w0, a[0]);
    not (w1, a[1]);
    not (w2, a[2]);
    not (w3, a[3]);

    // 9's complement: w + 1010
    // 1's complement: w + 0000

    // Full adder 0 (LSB)
    wire s0_p, c0_p, s0_q, c0_q;
    wire b0_p, b0_q;

    // For 9's comp adder (adder1)
    xor (b0_p, w0, 1'b0);        // b = 0 for LSB of 1010
    xor (s0_p, w0, 1'b0 ^ cin1); // sum = w ^ b ^ cin
    and (c0_p, w0, cin1);

    // For 1's comp adder (adder2)
    xor (b0_q, w0, 1'b0);        // b = 0
    xor (s0_q, w0, 1'b0 ^ cin2); // sum = w ^ b ^ cin
    and (c0_q, w0, cin2);

    // Full adder 1
    wire s1_p, c1_p, s1_q, c1_q;
    wire t1a, t1b, t1c, t1d;
    wire bp1, bq1;
    assign bp1 = 1'b1; // bit 1 of 1010 is 1
```

```
assign bq1 = 1'b0;

xor (t1a, w1, bp1);
xor (s1_p, t1a, c0_p);
and (t1b, w1, bp1);
and (t1c, t1a, c0_p);
or  (c1_p, t1b, t1c);

xor (t1a, w1, bq1);
xor (s1_q, t1a, c0_q);
and (t1b, w1, bq1);
and (t1c, t1a, c0_q);
or  (c1_q, t1b, t1c);

// Full adder 2
wire s2_p, c2_p, s2_q, c2_q;
wire bp2, bq2;
assign bp2 = 1'b0;  // bit 2 of 1010 is 0
assign bq2 = 1'b0;

xor (t1a, w2, bp2);
xor (s2_p, t1a, c1_p);
and (t1b, w2, bp2);
and (t1c, t1a, c1_p);
or  (c2_p, t1b, t1c);

xor (t1a, w2, bq2);
xor (s2_q, t1a, c1_q);
and (t1b, w2, bq2);
and (t1c, t1a, c1_q);
or  (c2_q, t1b, t1c);

// Full adder 3 (MSB)
wire s3_p, c3_p, s3_q, c3_q;
wire bp3, bq3;
assign bp3 = 1'b1;  // bit 3 of 1010 is 1
assign bq3 = 1'b0;

xor (t1a, w3, bp3);
xor (s3_p, t1a, c2_p);
and (t1b, w3, bp3);
and (t1c, t1a, c2_p);
or  (c3_p, t1b, t1c);

xor (t1a, w3, bq3);
xor (s3_q, t1a, c2_q);
and (t1b, w3, bq3);
and (t1c, t1a, c2_q);
or  (c3_q, t1b, t1c);

// 4-bit mux for output y = (r ? 9's comp : 1's comp)
wire nr;
not (nr, r);

// y[0]
wire t0a, t0b;
and (t0a, nr, s0_q);
and (t0b, r,  s0_p);
```

```
    or  (y[0], t0a, t0b);

    // y[1]
    and (t0a, nr, s1_q);
    and (t0b, r,  s1_p);
    or  (y[1], t0a, t0b);

    // y[2]
    and (t0a, nr, s2_q);
    and (t0b, r,  s2_p);
    or  (y[2], t0a, t0b);

    // y[3]
    and (t0a, nr, s3_q);
    and (t0b, r,  s3_p);
    or  (y[3], t0a, t0b);

endmodule
`timescale 1ns / 1ps

module rcomplement_tb;

    reg [3:0] a;
    reg cin1, cin2, r;
    wire [3:0] y;

    // Instantiate the Unit Under Test (UUT)
    rcomplement uut (
        .a(a),
        .cin1(cin1),
        .cin2(cin2),
        .r(r),
        .y(y)
    );

    integer i;

    initial begin
        $display("Time\tr\ta\ty\t(Complement)");
        $monitor("%4t\t%b\t%d\t%d\t%s", $time, r, a, y,
                 r ? "9's Comp" : "1's Comp");

        for (i = 1; i <= 9; i = i + 1) begin
            a = i;

            // Test for 1's complement
            r = 0;
            cin1 = 0;  // Not used
            cin2 = 0;  // No +1 for 1's comp
            #10;

            // Test for 9's complement
            r = 1;
            cin1 = 1;  // +1 for 9's comp
            cin2 = 0;  // Not used
            #10;
        end
```

```
        $finish;
    end

endmodule
```