-۱

برای حل این مسئله به شکل زیر عمل می‌کنیم:

```
module full_adder_one(
    output sum, cout,
    input in1, in2, cin
);
    wire w1,w2,w3;

    xor(w1,in1,in2);
    and(w2,w1,cin);
    and(w3,in1,in2);
    xor(sum,w1,cin);
    or(cout,w2,w3);
endmodule
```

```
module full_adder_four(
    output[3:0] sum,
    output cout,
    input[3:0] a,
    input[3:0] b,
    input cin
);
    wire c0,c1,c2;

    full_adder_one full_adder_one0(sum[0],c0,a[0],b[0],cin);
    full_adder_one full_adder_one1(sum[1],c1,a[1],b[1],c0);
    full_adder_one full_adder_one2(sum[2],c2,a[2],b[2],c1);
    full_adder_one full_adder_one3(sum[3],cout,a[3],b[3],c2);
endmodule


module bcd_one(
    output[3:0] sum,
    output cN,
    input[3:0] a,
    input[3:0] b,
    input cin
);
    wire[3:0] sumprime ;
    wire[3:0] temp;
    wire junk;
    supply0 gnd;
```

```
    wire carry_out;
    full_adder_four full_adder_four0(sumprime[3:0],cout,a[3:0],b[3:0],cin);

    assign cN = (sumprime[3]&sumprime[2])|(sumprime[3]&sumprime[1])|cout;

    assign temp[0] = 1'b0;
    assign temp[3] = 1'b0;
    assign temp[1] = cN;
    assign temp[2] = cN;


    full_adder_four full_adder_four1(sum[3:0],junk,sumprime[3:0],temp[3:0],gnd);




endmodule
```

```
module top(output[11:0] s,output cout, input[11:0] a, input[11:0] b , input cin);
    wire x0 , x1;

    bcd_one bcd_one0(s[3:0],x0,a[3:0],b[3:0],cin);
    bcd_one bcd_one1(s[7:4],x1,a[7:4],b[7:4],x0);
    bcd_one bcd_one2(s[11:8],cout,a[11:8],b[11:8],x1);

endmodule
```

تست‌بنچ :

```
reg cin;
wire[11:0] result;
wire cout;

top top0(result,cout,inputA,inputB,cin);

initial
    begin

    //inputA = 346 = 0011 0100 0110
    //inputB = 159  = 0001 0101 1001

    //Result = 505 = 0101 0000 0101
```

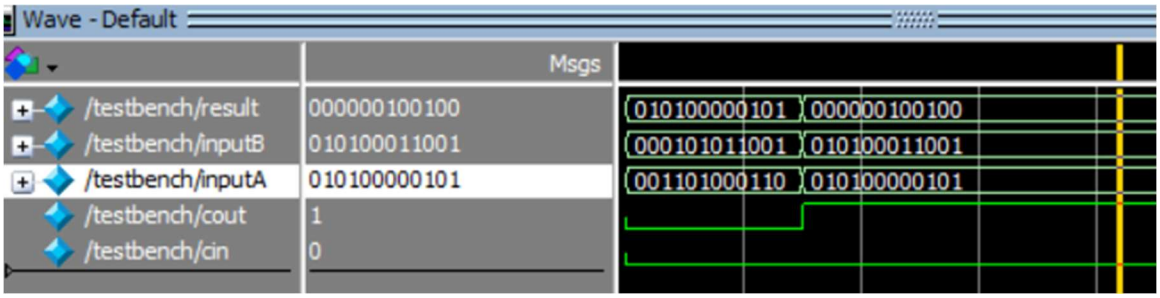
```
    cin = 0;
    inputA[11] = 0;
    inputA[10] = 0;
    inputA[9] = 1;
    inputA[8] = 1;
    inputA[7] = 0;
    inputA[6] = 1;
    inputA[5] = 0;
    inputA[4] = 0;
    inputA[3] = 0;
    inputA[2] = 1;
    inputA[1] = 1;
    inputA[0] = 0;


    inputB[11] = 0;
    inputB[10] = 0;
    inputB[9] = 0;
    inputB[8] = 1;
    inputB[7] = 0;
    inputB[6] = 1;
    inputB[5] = 0;
    inputB[4] = 1;
    inputB[3] = 1;
    inputB[2] = 0;
    inputB[1] = 0;
    inputB[0] = 1;
    #15;


    //inputA = 505 = 0101 0000 0101
    //inputB = 519 = 0101 0001 1001
    //Result = 1024 = 0001 0000 0010 0100 (Truth)
    //                  = (Cout=1) 0000 0010 0100 (Expected)

    cin = 0;
    inputA[11] = 0;
    inputA[10] = 1;
    inputA[9] = 0;
    inputA[8] = 1;
    inputA[7] = 0;
    inputA[6] = 0;
    inputA[5] = 0;
    inputA[4] = 0;
    inputA[3] = 0;
```

```
    inputA[2] = 1;
    inputA[1] = 0;
    inputA[0] = 1;


    inputB[11] = 0;
    inputB[10] = 1;
    inputB[9] = 0;
    inputB[8] = 1;
    inputB[7] = 0;
    inputB[6] = 0;
    inputB[5] = 0;
    inputB[4] = 1;
    inputB[3] = 1;
    inputB[2] = 0;
    inputB[1] = 0;
    inputB[0] = 1;
    #15;




    $finish;
end
endmodule
```

برای تخصیص مقادیر باینری(بی سی دی) از جدول زیر استفاده می‌کنیم :

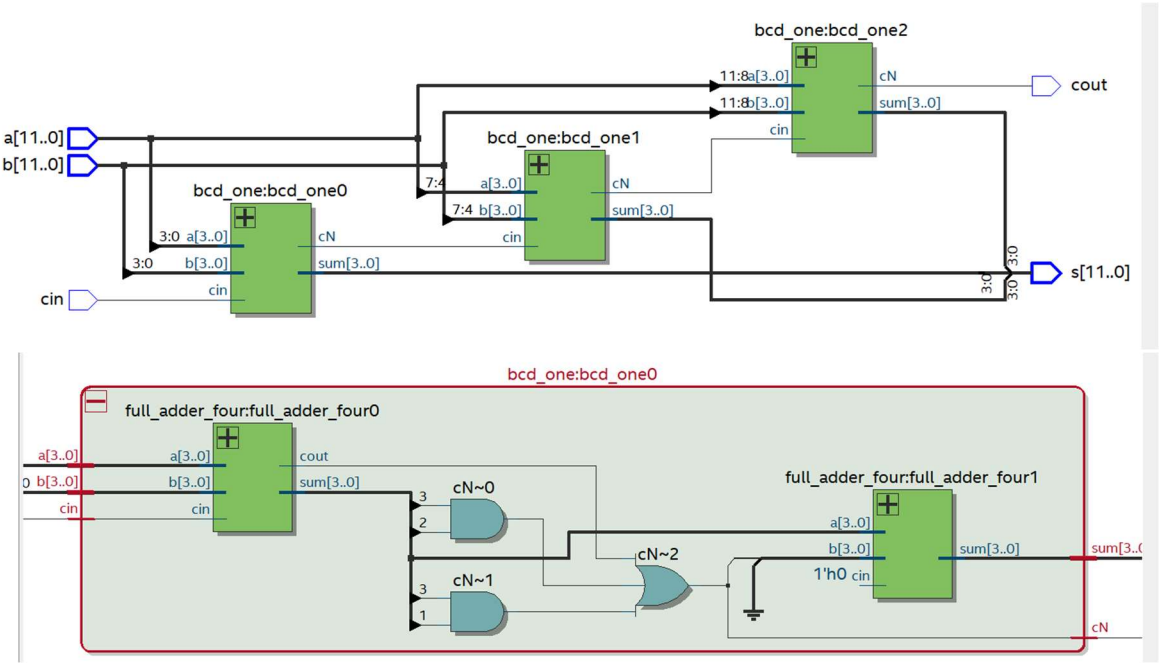
Binary Code A B C D	Decimal Number	BCD Code B_5 B_4 B_3 B_2 B_1
0 0 0 0	0	0 0 0 0 0
0 0 0 1	1	0 0 0 0 1
0 0 1 0	2	0 0 0 1 0
0 0 1 1	3	0 0 0 1 1
0 1 0 0	4	0 0 1 0 0
0 1 0 1	5	0 0 1 0 1
0 1 1 0	6	0 0 1 1 0
0 1 1 1	7	0 0 1 1 1
1 0 0 0	8	0 1 0 0 0
1 0 0 1	9	0 1 0 0 1

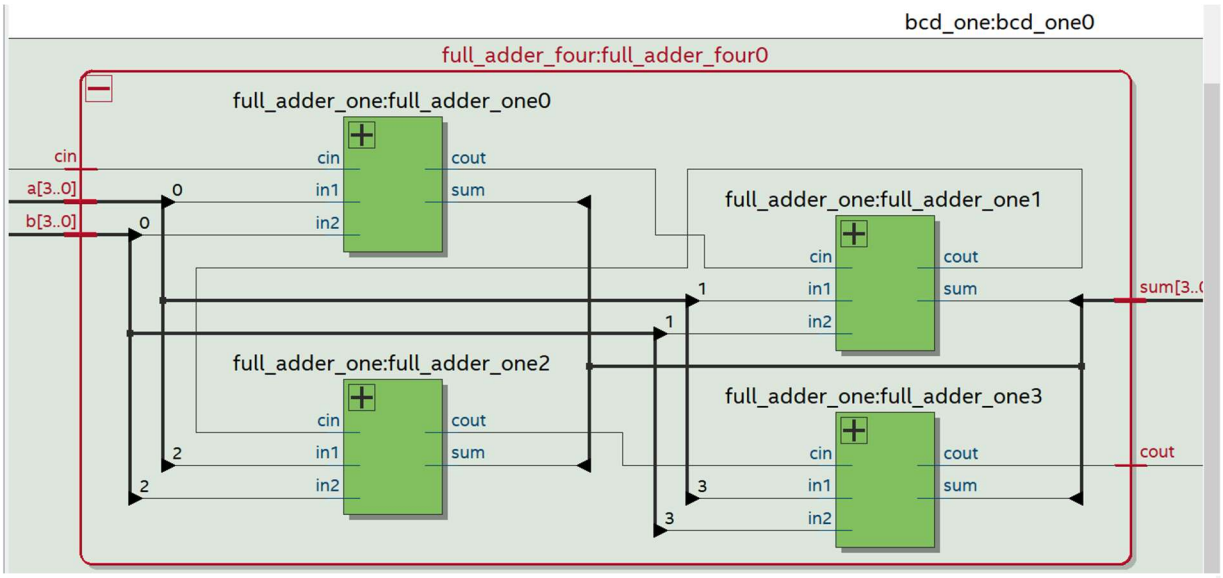
الان فقط کافیست سیگنال‌های Result شبیه‌سازی شده در مدل‌سیم را با تست‌بنچ مطابقت دهیم :

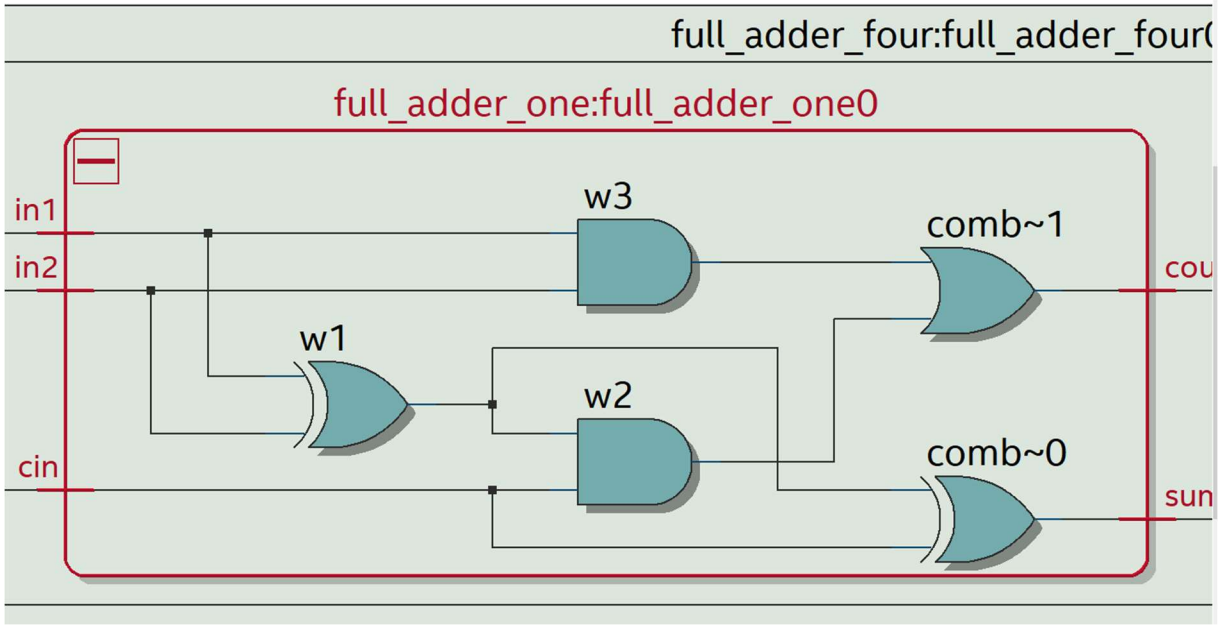


که کاملا مطابقت دارد!

شکل سنتزشده مدار هم به این شکل است:




bcd_one:bcd_one0
full_adder_four:full_adder_four0
full_adder_one:full_adder_one0
full_adder_one:full_adder_one1
full_adder_one:full_adder_one2
full_adder_one:full_adder_one3
cin
a[3..0]
b[3..0]
cout
sum
in1
in2
sum[3..
0
1
2
3


full_adder_four:full_adder_four(
full_adder_one:full_adder_one0
in1
in2
cin
w1
w2
w3
comb~0
comb~1
cou
sun

۲-برای حل مسئله زیر به شکل زیر عمل می‌کنیم : (clock و clear حساس به لبه بالارونده می‌باشند.)

```
module top(
    output[7:0] led,
    input clock,clear
    );

    reg Qd,Qc,Qb,Qa;
    reg ok;

    initial Qd = 1'b0;
    initial Qc = 1'b0;
    initial Qb = 1'b0;
    initial Qa = 1'b0;



    always @ (posedge clock or posedge clear)
    if (clear)
    begin
        Qa <= 1'b0;
        Qb <= 1'b0;
        Qc <= 1'b0;
        Qd <= 1'b0;
    end
    else
    begin
        Qa <= ~Qd;
        Qb <= Qa;
        Qc <= Qb;
        Qd <= Qc;
        ok <= 1'b1;
    end

    and(led[7],Qd,~Qc,~Qb,~Qa);
    and(led[6],Qd,Qc,~Qb,~Qa);
    and(led[5],Qd,Qc,Qb,~Qa);
    and(led[4],Qd,Qc,Qb,Qa);
    and(led[3],~Qd,Qc,Qb,Qa);
    and(led[2],~Qd,~Qc,Qb,Qa);
    and(led[1],~Qd,~Qc,~Qb,Qa);
    and(led[0],~Qd,~Qc,~Qb,~Qa);


endmodule
```

```
module testbench();

reg clk , clear;
wire[7:0] out;

top jj(out,clk,clear);

initial
  begin
    clk = 0;
    clear = 0;
 #80;
 clear = 1;
 #20;
 clear = 0;

  end




  always
    #20  clk =  ! clk;



endmodule
```

با استفاده از مدار کشیده‌شده در صورت سوال المان‌های مدار را به هم متصل می‌کنیم.

می‌دانیم Johnson Counter به ترتیب(از چپ به راست) سری زیر را تولید می‌کند : $(\overline{Q_d}Q_cQ_bQ_a)$

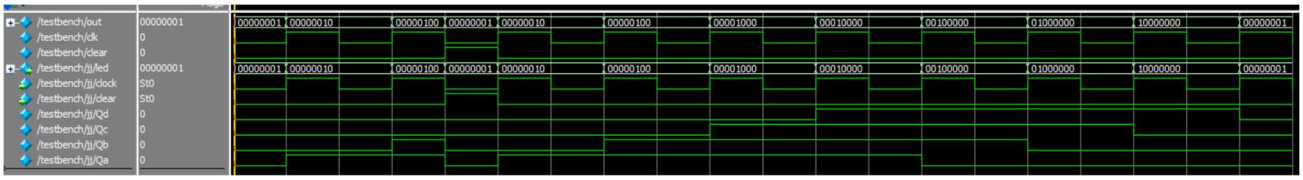
۰۰۰۰-۰۰۰۱-۰۰۱۱-۰۱۱۱-۱۱۱۱-۱۱۱۰-۱۱۰۰-۱۰۰۰

برای رسیدن به هدف اصلی یعنی ایجاد مدار تایمینگ (هر لحظه یک بیت روشن) می‌توانیم از گیت and استفاده کنیم.

به عنوان مثال در کلاک چهارم میخواهیم چهارمین بیت کم‌ارزش روشن شود.
از طرفی Johnson Counter در این حالت ۰۱۱۱ را تولید می‌کند؛ پس به سادگی میتوان با invert کردن Qd و and کردن با سایر بیت‌ها به خروجی مطلوب (۰۰۰۱۰۰۰۰) رسید.
واضح است که با این کار تداخلی پیش نمی‌آید و در هر حالت حتما یک بیت روشن می‌شود.

با clear کردن هم دستگاه به حالت اولیه $Q_dQ_cQ_bQ_a = 0000$ برمیگرد و خروجی ۰۰۰۰۰۰۰۱ را برمی‌گرداند.

شبیه‌سازی مدل‌سیم بر اساس تست‌بنچ :



شکل سنتزشده مدار هم به شکل زیر است: