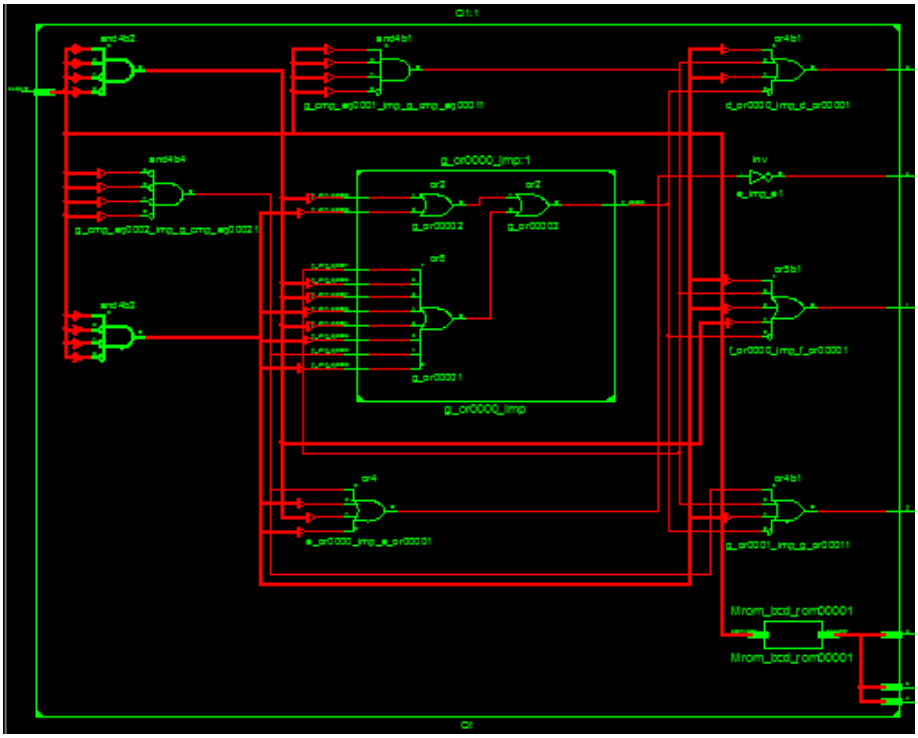


Figure 1: Zoomed Schematic

2 . Ans :

Verilog Code using Structural Style :

```
1
2  //---------------------------------STRUCTURAL STYLE-------------------------------------
3
4  /// ALU
5  ---------------------------------------
6  `timescale 1ns / 1ps
7  module ALU(A,B,C0,S1,S0,M,C_out,data_out
8  );
9  parameter N = 4;
10 parameter ZERO = 4'b0000;
11 input [N-1:0] A,B;
12 input S1,S0,M,C0;
13 output [N-1:0] data_out;
14 output C_out;
15
16 wire [N-1:0] AND,OR,XOR,XNOR,PASS,SUM,DIFF1,DIFF2;
17 wire cand,cor,cxor,cxnor,cpass,csum,cdiff1,cdiff2;
```

```
wire [N-1:0] comp_A,comp_B;

andop M0(A,B,AND);
assign cand = 1'bz;
orop M1(A,B,OR);
assign cor = 1'bz;
xorop M2(A,B,XOR);
assign cxor = 1'bz;
xnorop M3(A,B,XNOR);
assign cxnor = 1'bz;
ripple_adder A0(A,ZERO,C0,cpass,PASS);
ripple_adder A1(A,B,C0,csum,SUM);
compop CB(B,comp_B);
ripple_adder A3(A,comp_B,C0,cdiff1,DIFF1);
compop CA(A,comp_A);
ripple_adder A4(comp_A,B,C0,cdiff2,DIFF2);

mux8to1 M81({cand,AND},{cor,OR},{cxor,XOR},{cxnor,XNOR},{cpass,PASS},{csum,SUM},{cdiff1,DIFF1},{cdiff

endmodule

// AND MODULE
---------------------------------------
`timescale 1ns / 1ps
module andop(A,B,C
);
parameter N=4;
input [N-1:0] A,B;
output [N-1:0] C;
genvar p;

generate
for(p = 0; p<N; p = p+1)
begin : and1p
and AG(C[p],A[p],B[p]);
end
endgenerate

endmodule
// OR MODULE
---------------------------------------
`timescale 1ns / 1ps
module orop(A,B,C
);
parameter N=4;
input [N-1:0] A,B;
output [N-1:0] C;
genvar p;

generate
for(p = 0; p<N; p = p+1)
begin : or1p
or OG(C[p],A[p],B[p]);
end
endgenerate


endmodule
// XOR MODULE
```

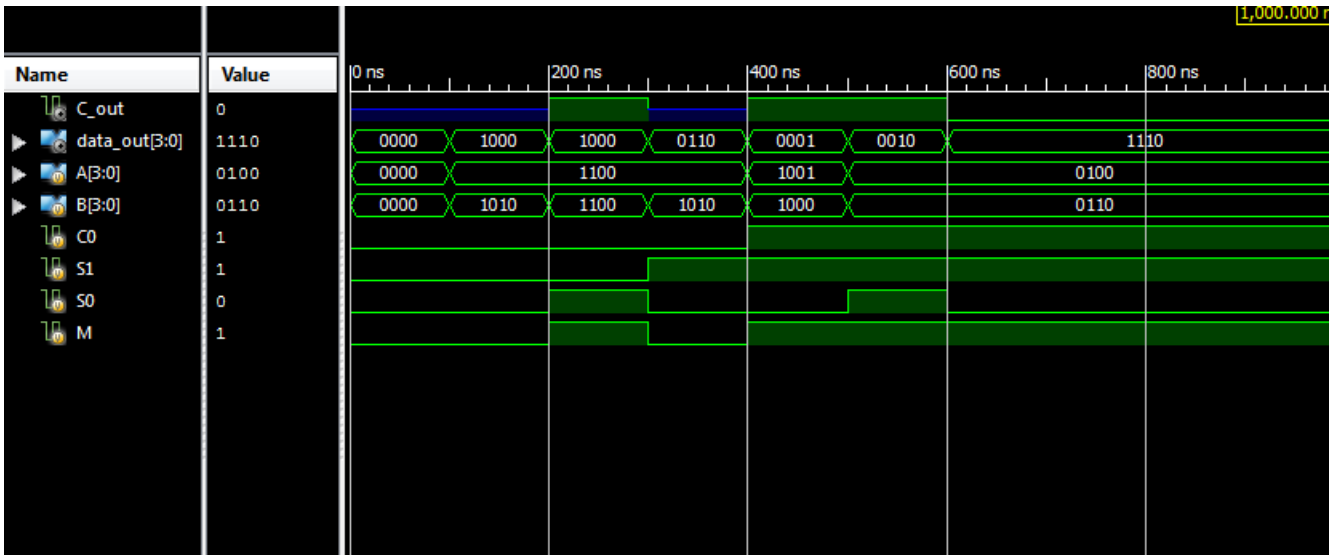
```
77  ----------------------------------------
78  `timescale 1ns / 1ps
79  module xorop(A,B,C
80  );
81  parameter N=4;
82  input [N-1:0] A,B;
83  output [N-1:0] C;
84  genvar p;
85
86  generate
87  for(p = 0; p<N; p = p+1)
88  begin : xor1p
89  xor XG(C[p],A[p],B[p]);
90  end
91  endgenerate
92
93
94  endmodule
95  //XNOR MODULE
96  ----------------------------------------
97  `timescale 1ns / 1ps
98  module xnorop(A,B,C
99  );
100 parameter N=4;
101 input [N-1:0] A,B;
102 output [N-1:0] C;
103 genvar p;
104
105 generate
106 for(p = 0; p<N; p = p+1)
107 begin : xn1p
108 xnor XNG(C[p],A[p],B[p]);
109 end
110 endgenerate
111
112
113 endmodule
114
115 //COMPLEMENT MODULE
116 ----------------------------------------
117 `timescale 1ns / 1ps
118 module compop(A,C
119 );
120 parameter N=4;
121 input [N-1:0] A;
122 output [N-1:0] C;
123 genvar p;
124
125 generate
126 for(p = 0; p<N; p = p+1)
127 begin : comp1p
128 not NG(C[p],A[p]);
129 end
130 endgenerate
131
132
133 endmodule
134
135 // FULL ADDER
```

```
136 ---------------------------------------
137 `timescale 1ns / 1ps
138 module full_adder(A,B,Cin,Cout,S
139 );
140 input A,B,Cin;
141 output Cout,S;
142
143 wire [2:0] w;
144
145 xor X1(w[0],A,B), X2(S,w[0],Cin);
146 and A1(w[1],A,B), A2(w[2],w[0],Cin);
147 or R1(Cout,w[1],w[2]);
148
149 endmodule
150
151 // RIPPLE ADDER
152 ---------------------------------------
153 `timescale 1ns / 1ps
154 module ripple_adder(A,B,Cin,Cout,S
155 );
156 parameter N = 4;
157 input [N-1:0] A,B;
158 input Cin;
159 output [N-1:0] S;
160 output Cout;
161 wire [N:0] carry;
162 assign carry[0] = Cin;
163 assign Cout = carry[N];
164
165 genvar p;
166 generate
167 for(p=0;p<N;p=p+1)
168 begin : fa_loop
169 full_adder FA(A[p],B[p],carry[p],carry[p+1],S[p]);
170 end
171 endgenerate
172
173
174 endmodule
175
176 // MUX 2X1
177 ---------------------------------------
178 `timescale 1ns / 1ps
179 module mux2to1(in0,in1,sel,out
180 );
181 parameter N = 5;
182 input [N-1:0] in1,in0;
183 input sel;
184 output [N-1:0] out;
185
186 assign out = (sel == 1'b0)? in0 : in1;
187
188
189 endmodule
190
191 //MUX 4X1
192 ---------------------------------------
193 `timescale 1ns / 1ps
194 module mux4to1(in0,in1,in2,in3,sel,out
```

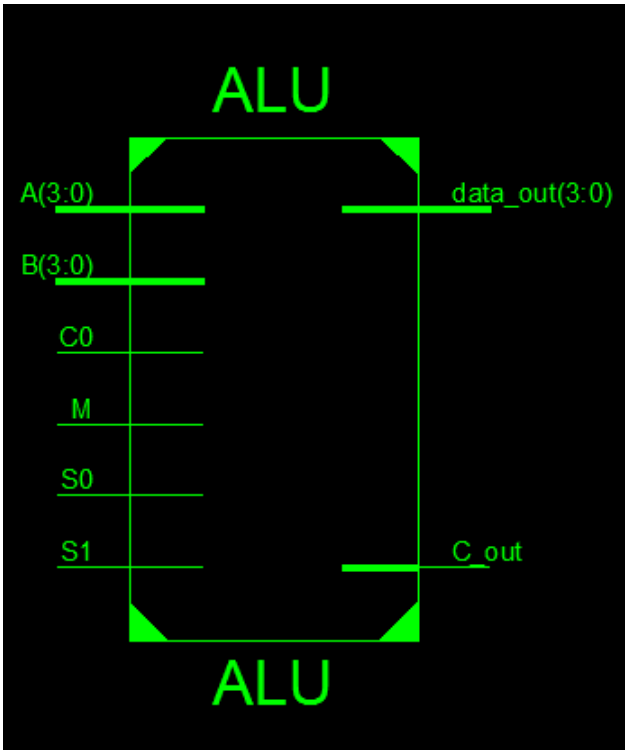
```
);
parameter N = 5;
input [N-1:0] in3,in2,in1,in0;
input [1:0] sel;
output [N-1:0] out;
wire [N-1:0] t1,t0;

mux2to1 M0(in0,in1,sel[0],t0);
mux2to1 M1(in2,in3,sel[0],t1);
mux2to1 M2(t0,t1,sel[1],out);

endmodule

// MUX 8X1
----------------------------------------
`timescale 1ns / 1ps
module mux8to1(in0,in1,in2,in3,in4,in5,in6,in7,sel2,sel1,sel0,out
);

parameter N = 5;
input [N-1:0] in0,in1,in2,in3,in4,in5,in6,in7;
input sel2,sel1,sel0;
output [N-1:0] out;
wire [N-1:0] t1,t0;

mux4to1 M0(in0,in1,in2,in3,{sel1,sel0},t0);
mux4to1 M1(in4,in5,in6,in7,{sel1,sel0},t1);
mux2to1 M2(t0,t1,sel2,out);

endmodule

// ---------------------------------TEST BENCH-------------------------
`timescale 1ns / 1ps
module TEST;

// Inputs
reg [3:0] A;
reg [3:0] B;
reg C0;
reg S1;
reg S0;
reg M;

// Outputs
wire C_out;
wire [3:0] data_out;

// Instantiate the Unit Under Test (UUT)
ALU uut (
.A(A),
.B(B),
.C0(C0),
.S1(S1),
.S0(S0),
.M(M),
.C_out(C_out),
.data_out(data_out)
);

```

```
254 initial begin
255 // Initialize Inputs
256 A = 0;
257 B = 0;
258 C0 = 0;
259 S1 = 0;
260 S0 = 0;
261 M = 0;
262
263 // Wait 100 ns for global reset to finish
264 #100;
265 A = 4'b1100;
266 B = 4'b1010;
267 C0 = 0;
268 S1 = 0;
269 S0 = 0;
270 M = 0;
271 #100
272 A = 4'b1100;
273 B = 4'b1100;
274 C0 = 0;
275 S1 = 0;
276 S0 = 1;
277 M = 1;
278 #100;
279 A = 4'b1100;
280 B = 4'b1010;
281 C0 = 0;
282 S1 = 1;
283 S0 = 0;
284 M = 0;
285 #100
286 A = 4'b1001;
287 B = 4'b1000;
288 C0 = 1;
289 S1 = 1;
290 S0 = 0;
291 M = 1;
292 #100
293 A = 4'b0100;
294 B = 4'b0110;
295 C0 = 1;
296 S1 = 1;
297 S0 = 1;
298 M = 1;
299 #100
300 A = 4'b0100;
301 B = 4'b0110;
302 C0 = 1;
303 S1 = 1;
304 S0 = 0;
305 M = 1;
306
307 // Add stimulus here
308
309 end
310
311 endmodule
```

Simulation Results :



Schematic :



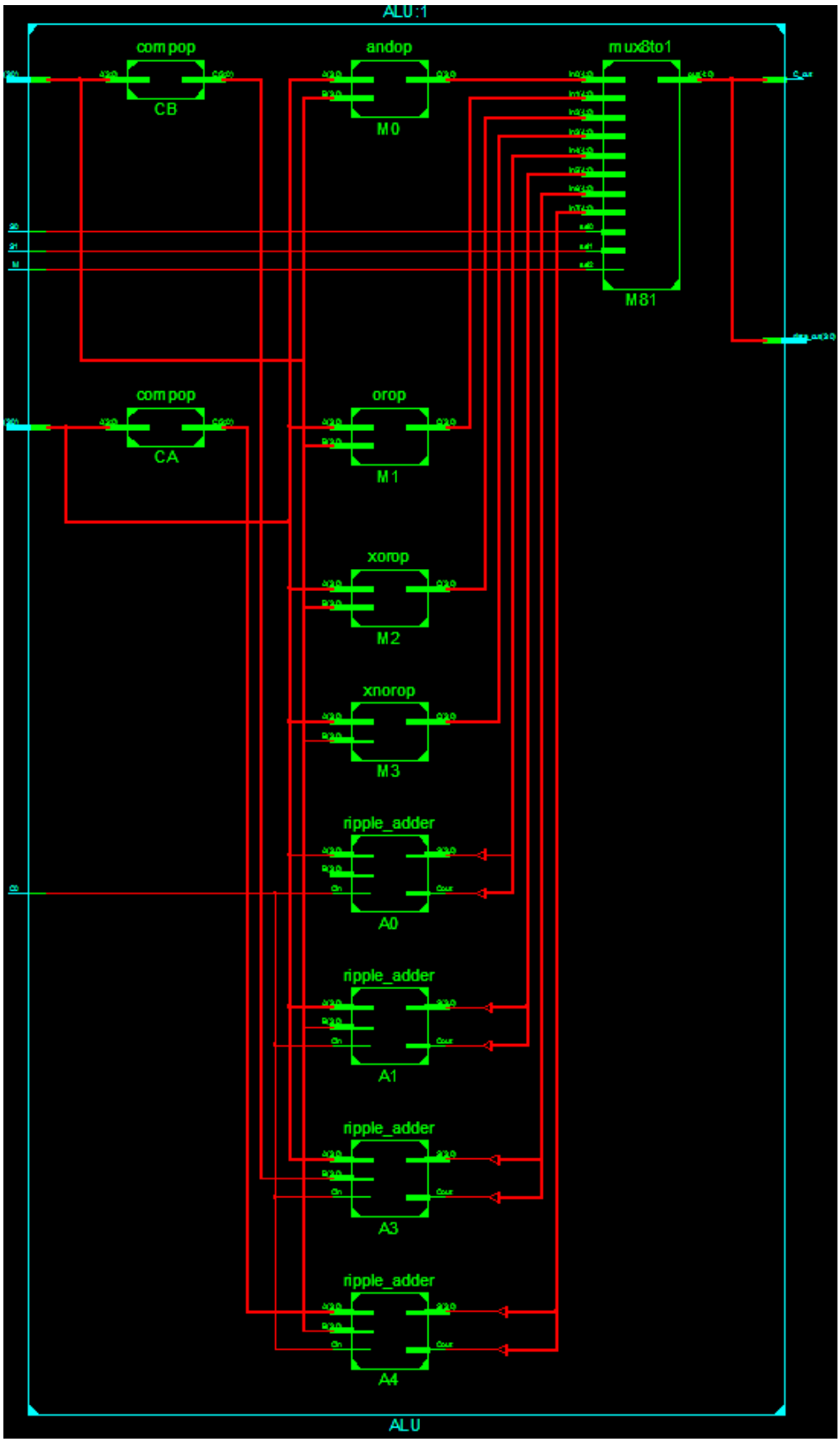


Figure 2: Block Level Schematic

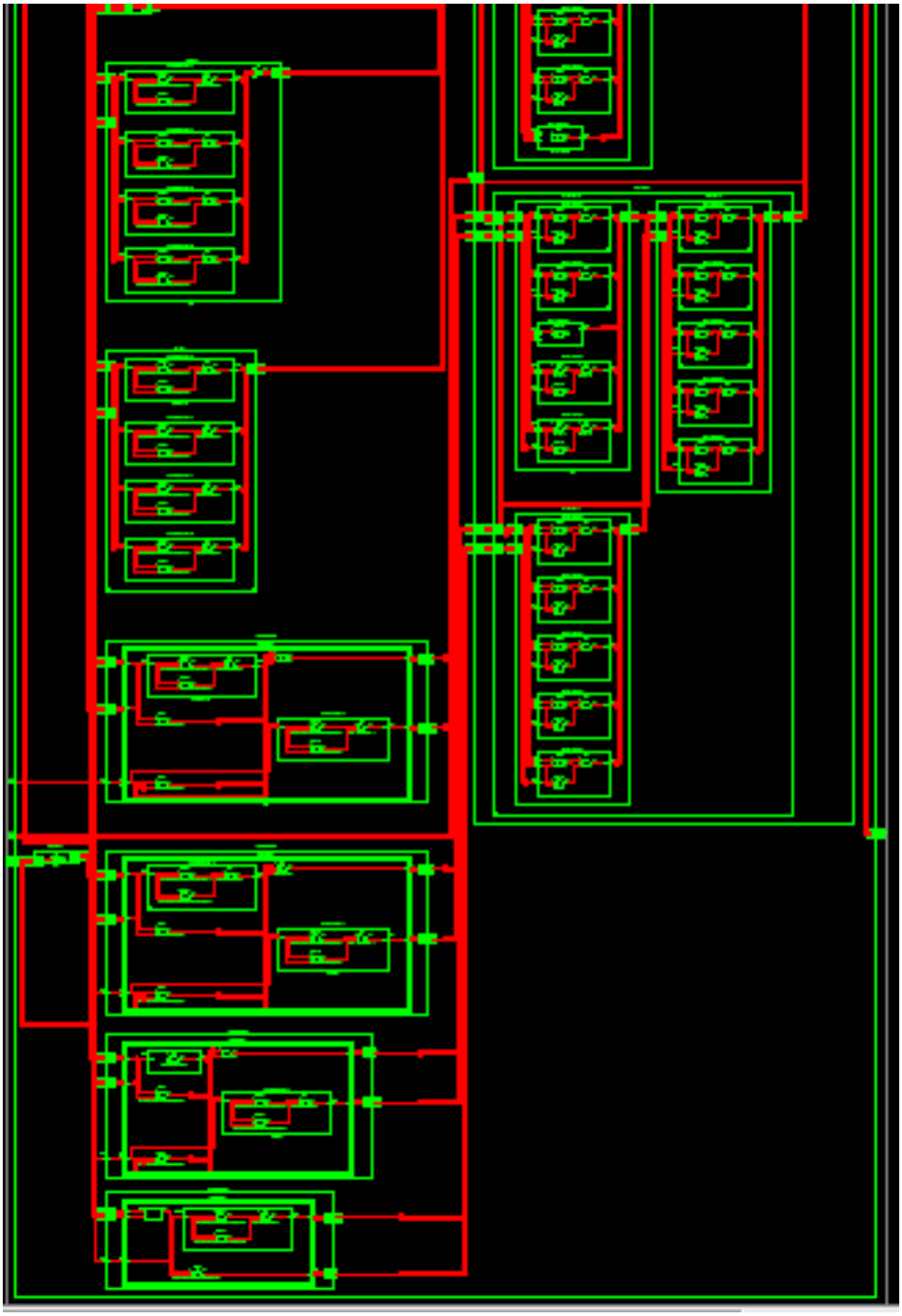


Figure 3: Fully Blown out Schematic

Remarks

- Please note that that the output can be followed by the bcd to 7 segment converter for displaying the output on a 7 segment display.
- The remaining user constraints can be added, for displaying the output as mentioned and it can be implemented on FPGA.

3 . Ans :

Verilog Code using Structural Style :

```
//------------------------------------------------------------------------------
//--------------------------Carry Save Adder-----------------------------------
//--------------(with ripple carry adder in final stage)----------------------


//Full adder-----------------------------------------------
`timescale 1ns / 1ps
module full_adder(A,B,Cin,S,Cout
);
input A,B,Cin;
output S,Cout;

wire [2:0] w;

xor X1(w[0],A,B),X2(S,Cin,w[0]);
and A1(w[1],A,B),A2(w[2],w[0],Cin);
or O1(Cout,w[1],w[2]);

endmodule


//Carry Save Adder-----------------------------------------

`timescale 1ns / 1ps
module carry_save_adder(A,B,S,Cout
);
parameter N = 8;
parameter ZERO = 8'b00000000;
input [N-1:0] A,B;
output [N-1:0] S;
wire [N:0] S2;
output Cout;

wire [N:0] S1,C1;
wire [N+1:0] C;
assign S = S2[N-1:0];
assign C[0] = 1'b0;
```