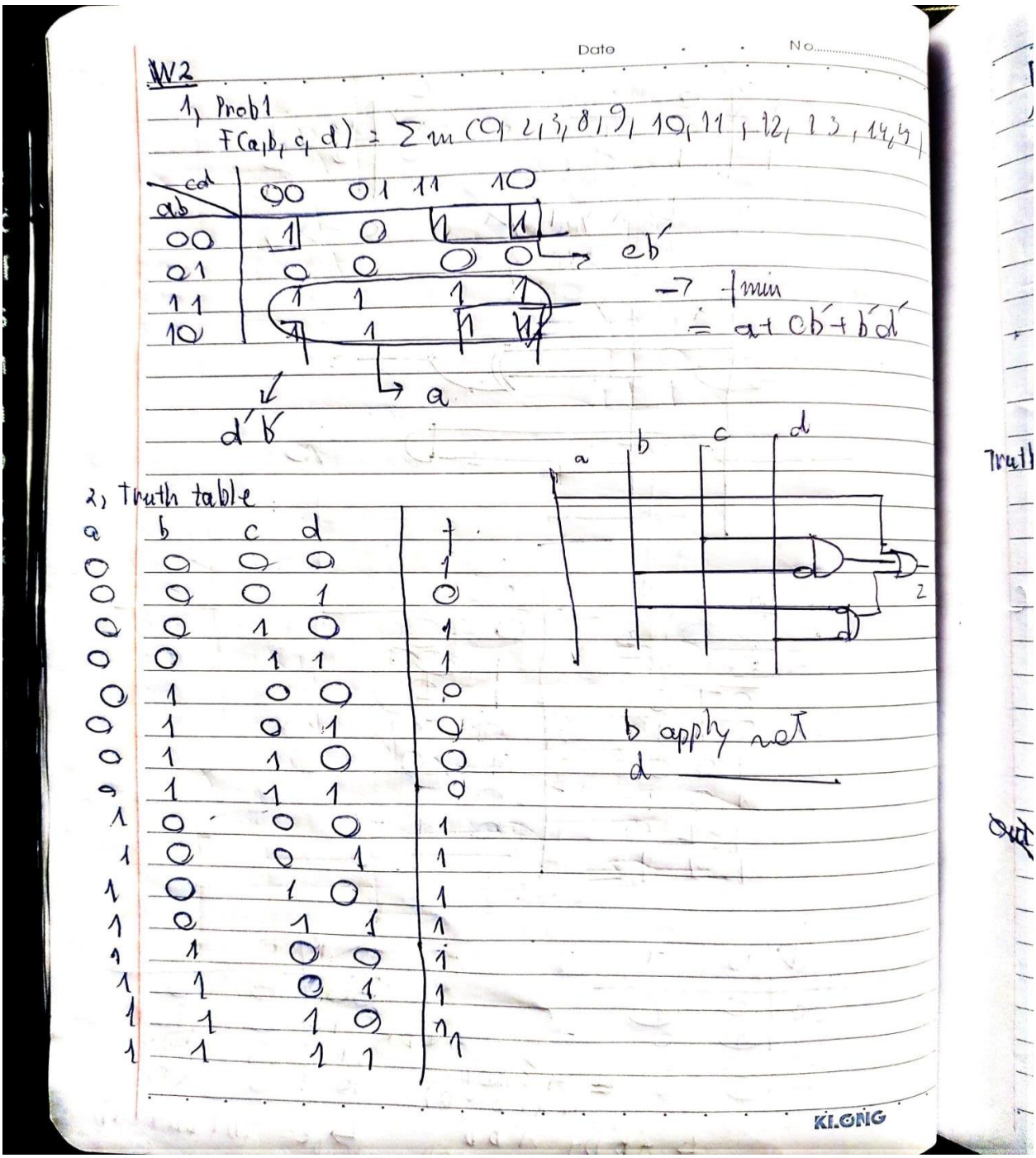
Week 2 Exercise – Dang Tran Nhat Minh – 20193231

Ex1:

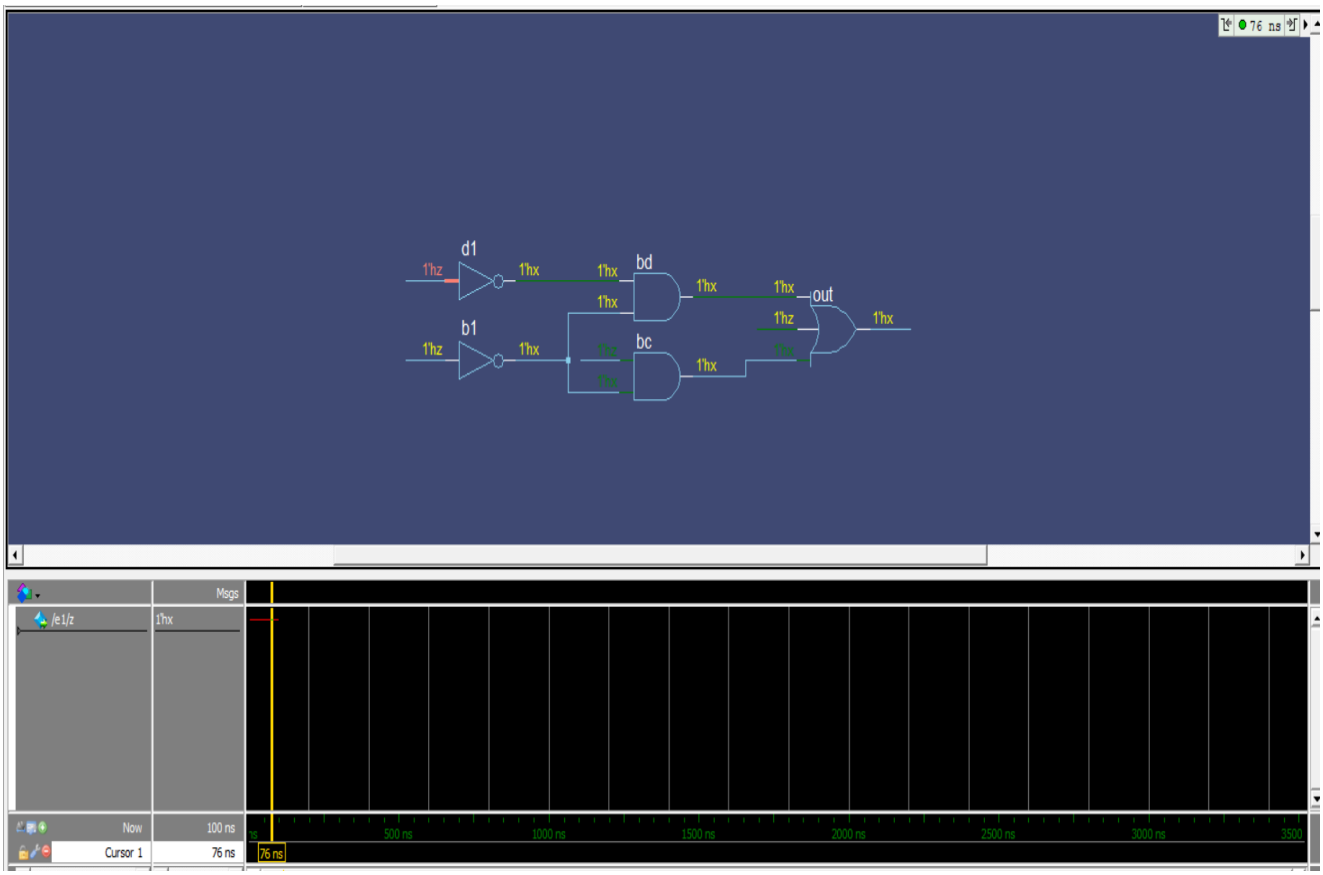
K-map



Verilog code

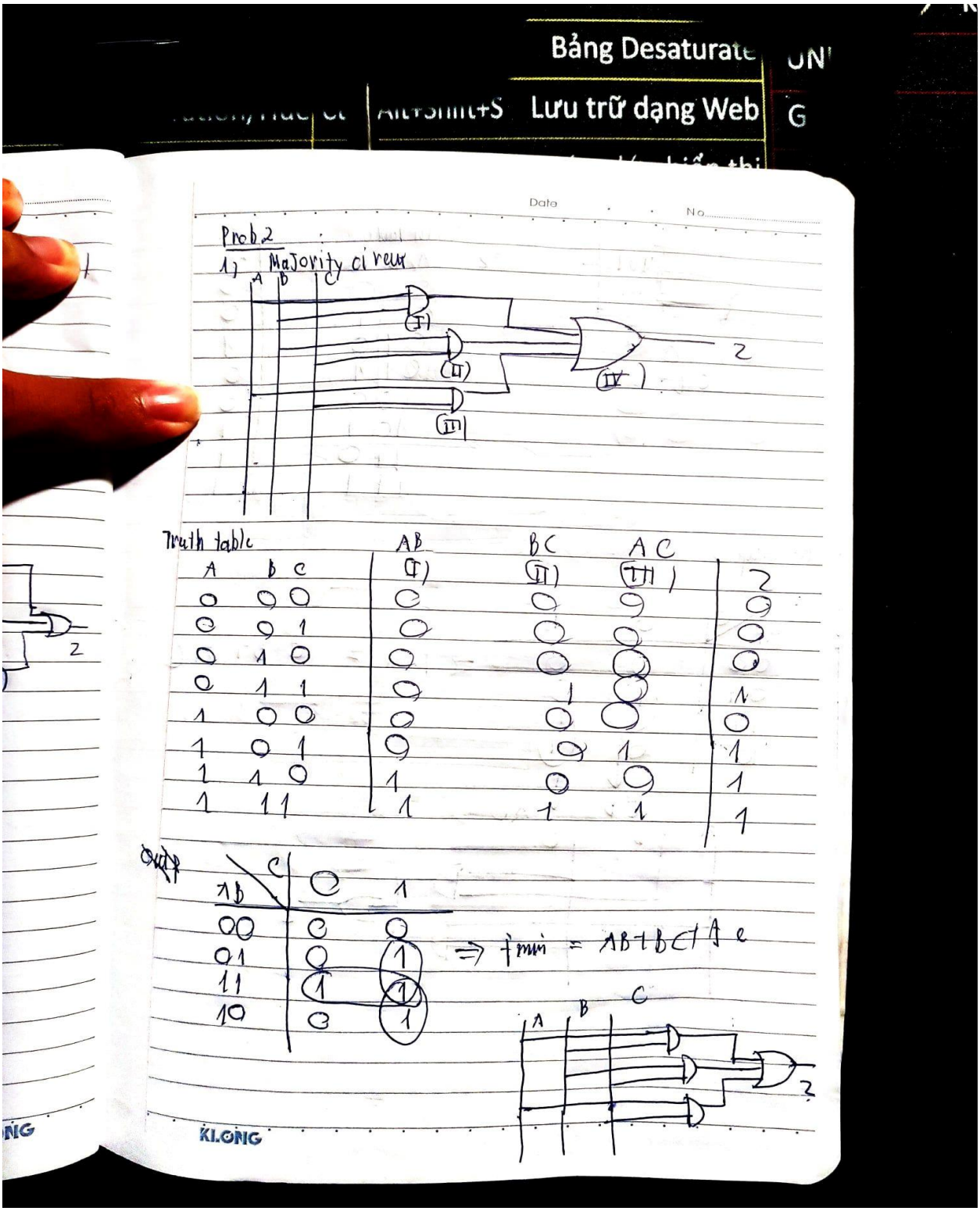
```
module e1(a,b,c,d,z);
input a,b,c,d;
output z;
wire n1,n2,n3,n4;
// b with NOT gate
not b1(n1,b);
// d with NOT gate
not d1(n2,d);
// b NOT with c
and bc(n3,n1,c);
// b NOT with d NOT
and bd(n4,n1,n2);
//Output
or out(z,a,n3,n4);

endmodule;
```



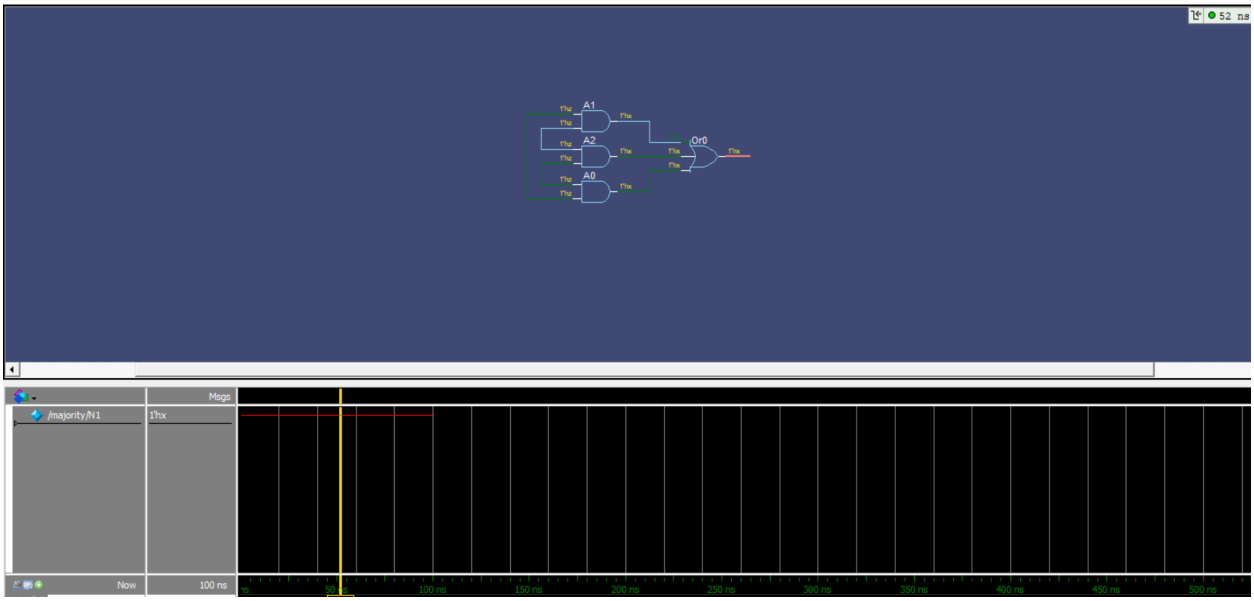
Ex2:

Majority circuit

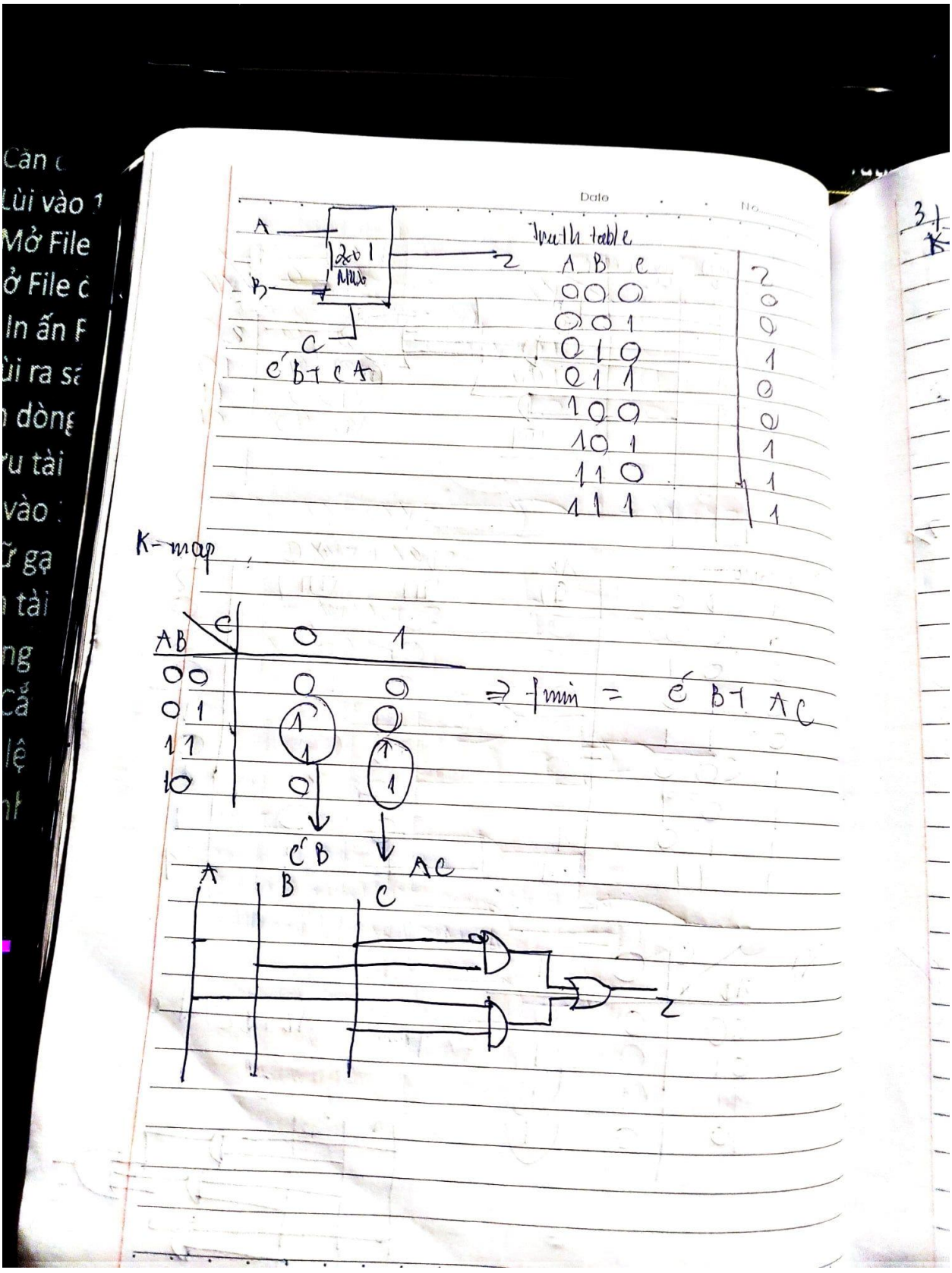


Verilog code

```
1  module majority(a,b,c,z);
2  output z;
3  input a,b,c;
4  wire n1,n2,n3;
5
6  and A0(N1,a,b),
7      A1(N2,b,c),
8      A2(N3,a,c);
9  or Or0(z,N1,N2,N3);
10 endmodule
11
12
```



MUX 2x1

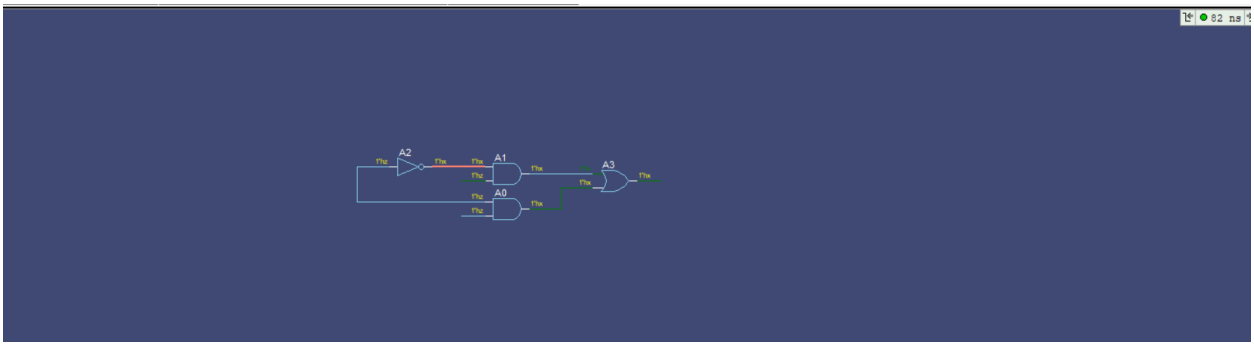


Verilog code

C:/modeltech64_10.7/examples/mux.v - Default *

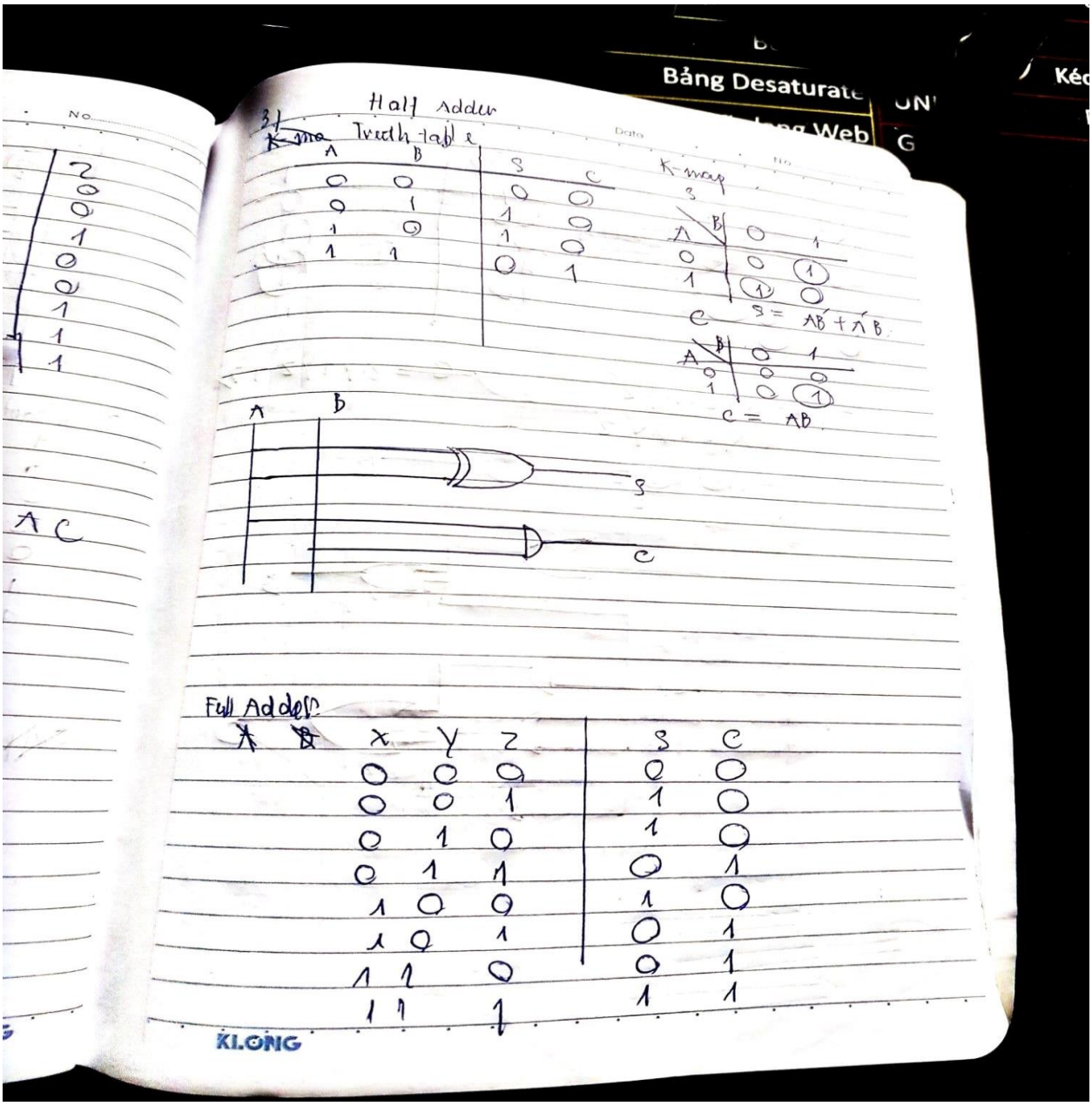
Ln#

```
1  module mux21(z,a,b,c);
2  output z;
3  input a,b,c;
4  wire n1,n2,n3;
5
6  and A0(n1,a,c),
7      A1(n2,b,n2);
8  not A2(n2,c);
9  or A3(n3,n1,n2);
10
11 endmodule
12
```



Ex 3:

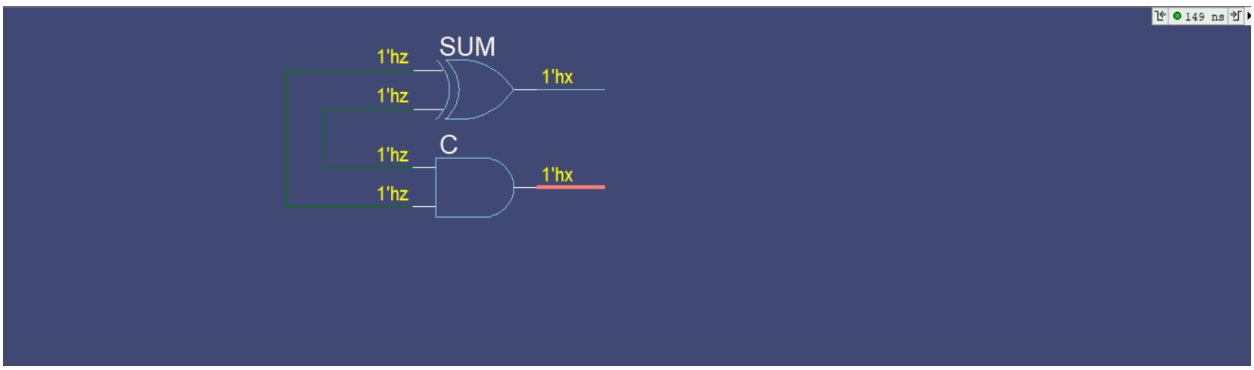
3.1 Half adder



Verilog code

C:/modeltech64_10.7/examples/HalfAdder.v - Default *

```
Ln#
1  module half_add(X,Y,Z,C);
2  input X,Y;
3  output Z,C;
4  xor SUM(S,X,Y);
5  and CARRY(C,X,Y);
6  endmodule
7
```



3.2 Full adder

Date / / No.

S

xy \ z	0	1
00	0	(1)
01	(1)	0
11	0	(1)
10	(1)	0

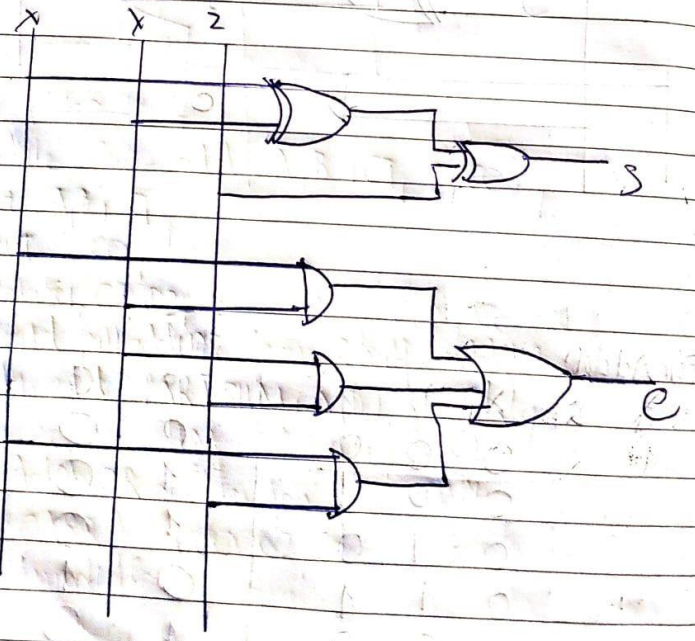
C

xy \ z	0	1
00	0	0
01	0	(1)
11	(1)	(1)
10	0	(1)

$\Rightarrow S = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xyz + x\bar{y}\bar{z}$

$= x \oplus y \oplus z$

$C = xy + yz + zx$



KI.ONG

Ripple C

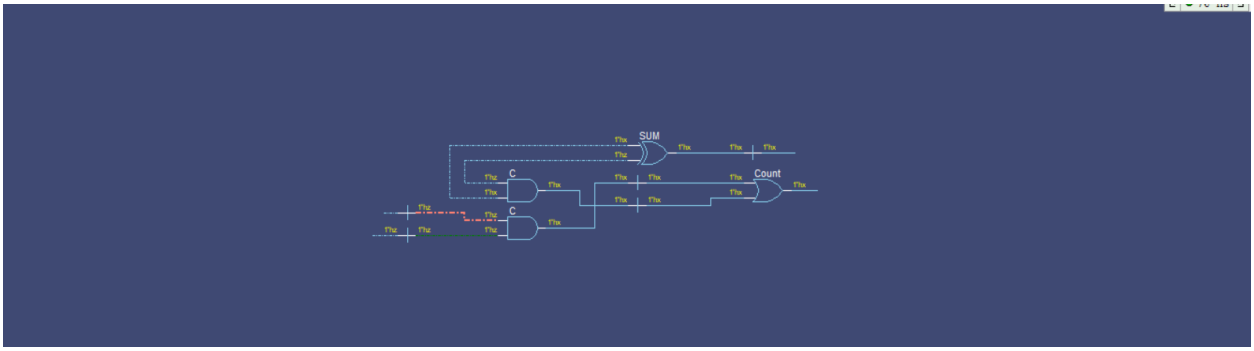
C16

Verilog code

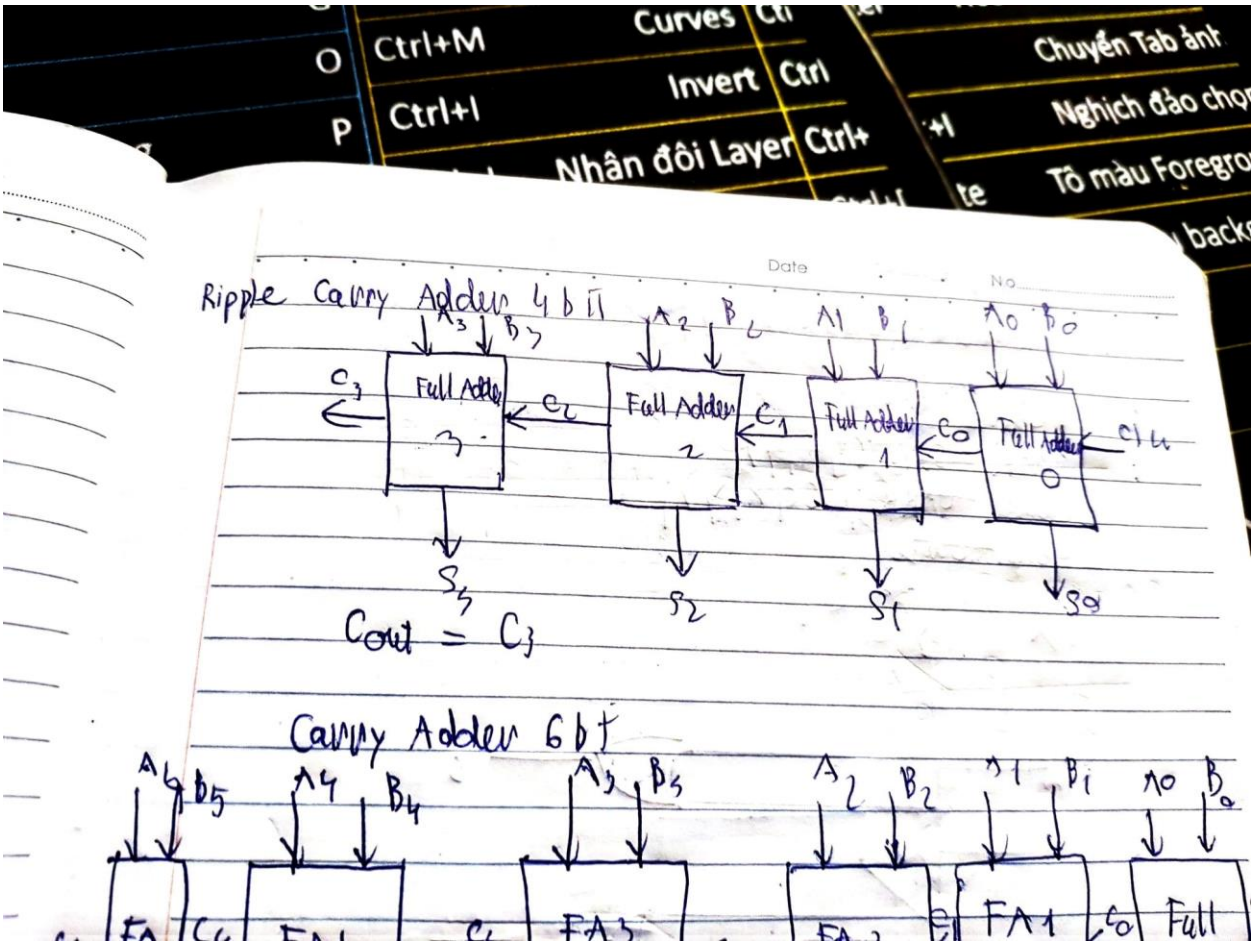
C:/modeltech64_10.7/examples/fulladder.v - Default *

Ln#

```
1   // Full Adder
2   module full_add(A,B,Ci,S,Co);
3   input A,B,Ci;
4   output S,Co;
5   wire S1,C1,C2;
6   // Build full adder from 2 half-adders
7   half_add PARTSUM (A,B,S1,C2),
8             SUM (S1,Ci,S,C2);
9    or CARRY (Co,C2,C1);
10  endmodule
11
```



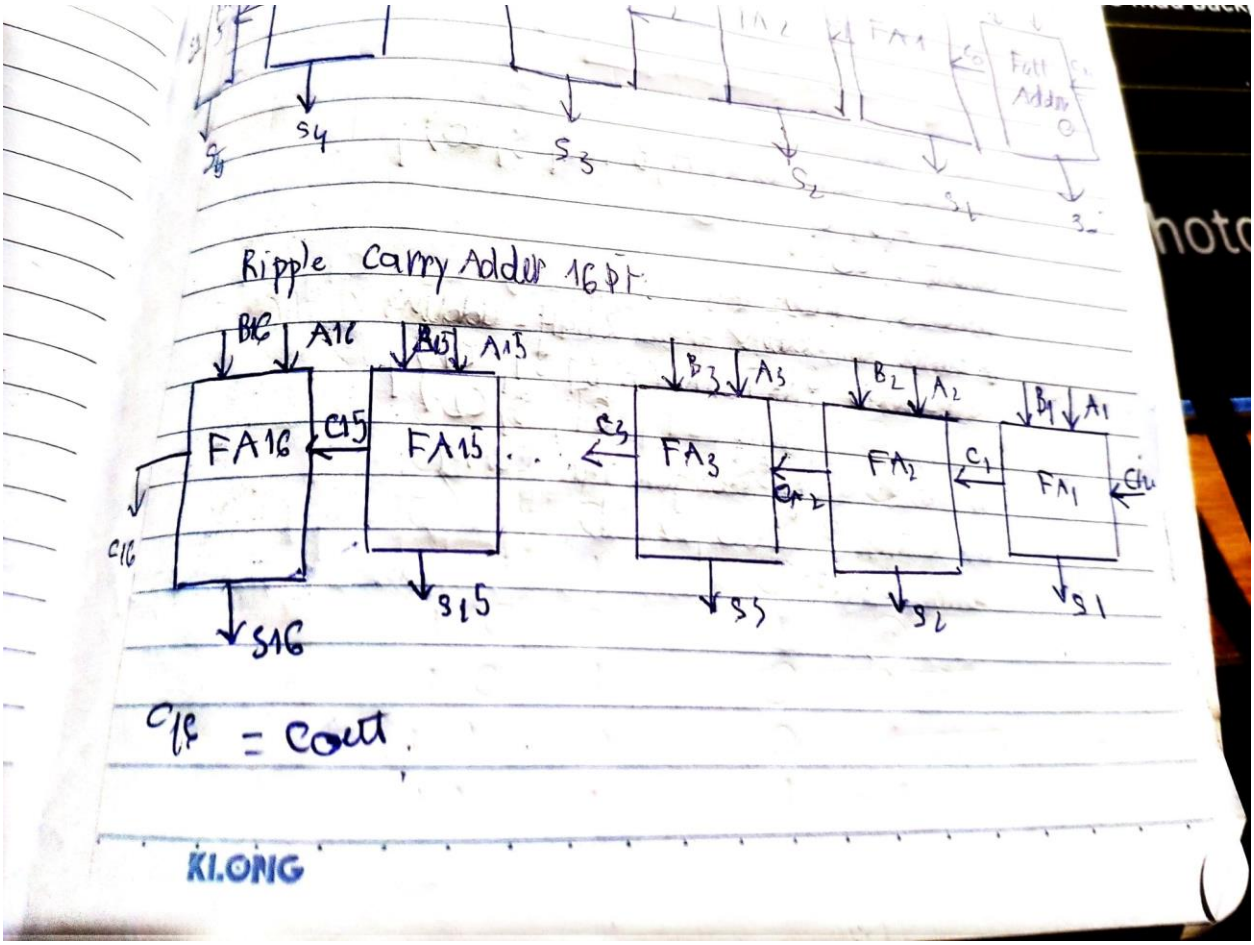
3.3 Ripple_Carry_Adder_4bit



Verilog code

```
1
2   // Ex 3
3   // Half adder
4   module half_add(X,Y,Z,C);
5   input X,Y;
6   output Z,C;
7   xor SUM(S,X,Y);
8   and CARRY(C,X,Y);
9   endmodule
10
11  // Full Adder
12  module full_adder(A,B,Ci,S,Co);
13  input A,B,Ci;
14  output S,Co;
15  wire S1,C1,C2;
16  // Build full adder from 2 half-adders
17  half_add PARTSUM (A,B,S1,C2),
18           SUM (S1,Ci,S,C2);
19   or CARRY (Co,C2,C1);
20  endmodule
21
22   // Ripple carry adder 4 bit
23  module rp_cr_adder_4bit(a,b,o,cout);
24  input[3:0] a;
25  input[3:0] b;
26  output[3:0] o;
27  output cout;
28  wire n1,n2,n3;
29  full_adder fa0(a[0],b[0],0,o[0],n1);
30  full_adder fa1(a[1],b[1],n1,o[1],n2);
31  full_adder fa2(a[2],b[2],n2,o[2],n3);
32  full_adder fa3(a[3],b[3],n3,o[3],cout);
33  endmodule
34
```

3.4 Ripple_carry_adder_16bit



Verilog code

```
Ln#
 1
 2  // Ex 3
 3  // Half adder
 4  module half_add(X,Y,Z,C);
 5  input X,Y;
 6  output Z,C;
 7  xor SUM(S,X,Y);
 8  and CARRY(C,X,Y);
 9  endmodule
10
11  // Full Adder
12  module full_adder(A,B,Ci,S,Co);
13  input A,B,Ci;
14  output S,Co;
15  wire S1,C1,C2;
16  // Build full adder from 2 half-adders
17  half_add PARTSUM (A,B,S1,C2),
18          SUM (S1,Ci,S,C2);
19   or CARRY (Co,C2,C1);
20  endmodule
21
22  // Ripple carry adder 16 bit
23  module rp_16(s,cout,a,b,cin);
24  input [15:0] a,b;
25  input cin;
26  output [15:0] s;
27  output cout;
28  wire [14:0] c;
29  full_adder FA1(s[0],c[0],a[0],b[0],cin);
30  full_adder FA2(s[1],c[1],a[1],b[1],c[0]);
31  full_adder FA3(s[2],c[2],a[2],b[2],c[1]);
32  full_adder FA4(s[3],c[3],a[3],b[3],c[2]);
33  full_adder FA5(s[4],c[4],a[4],b[4],c[3]);
34  full_adder FA6(s[5],c[5],a[5],b[5],c[4]);
35  full_adder FA7(s[6],c[6],a[6],b[6],c[5]);
36  full_adder FA8(s[7],c[7],a[7],b[7],c[6]);
37  full_adder FA9(s[8],c[8],a[8],b[8],c[7]);
38  full_adder FA10(s[9],c[9],a[9],b[9],c[8]);
39  full_adder FA11(s[10],c[10],a[10],b[10],c[9]);
40  full_adder FA12(s[11],c[11],a[11],b[11],c[10]);
41  full_adder FA13(s[12],c[12],a[12],b[12],c[11]);
42  full_adder FA14(s[13],c[13],a[13],b[13],c[12]);
43  full_adder FA15(s[14],c[14],a[14],b[14],c[13]);
44  full_adder FA16(s[15],cout,a[15],b[15],c[14]);
45  endmodule
46
```