Zübeyir Bodur 20.10.2019

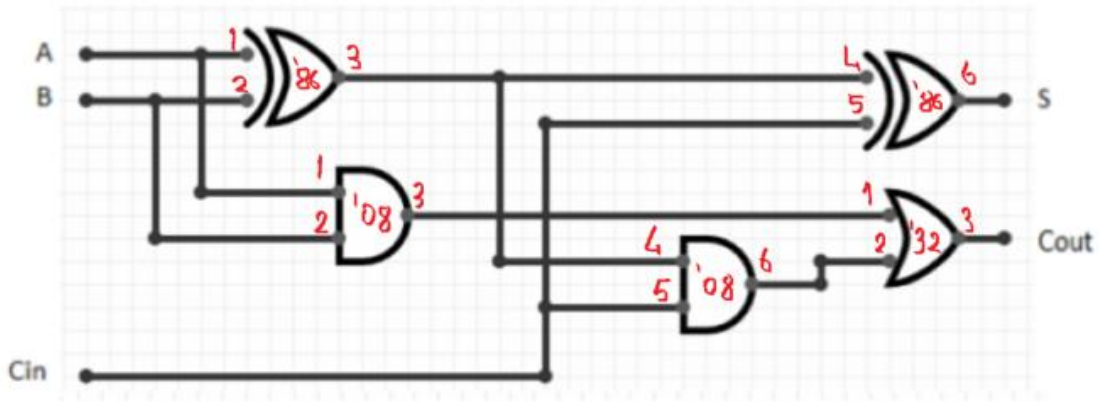
21702382

CS-223-02

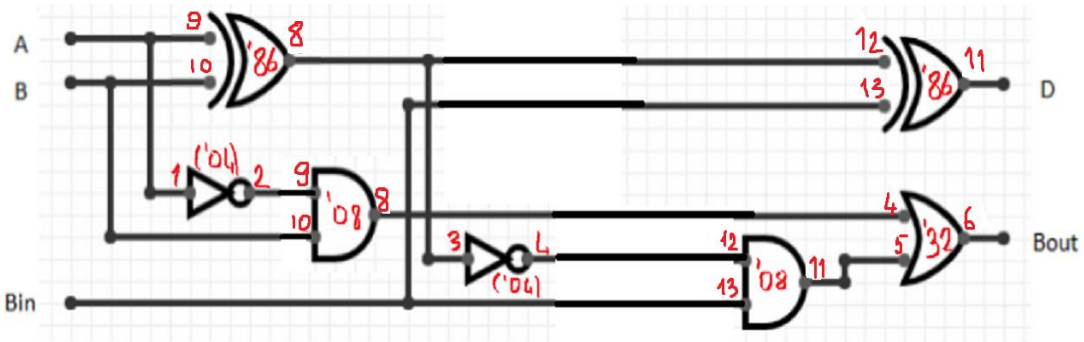
Lab-02

PRELIMINARY WORK

1. 1-Bit Full-Adder & Subtractor



Schematic for 1-bit full-adder



Schematic for 1-bit full-subtractor

Inputs			Full Adder		Full Subtractor	
A	B	Cin / Bin	Sum	Carry	Diff.	Borrow
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

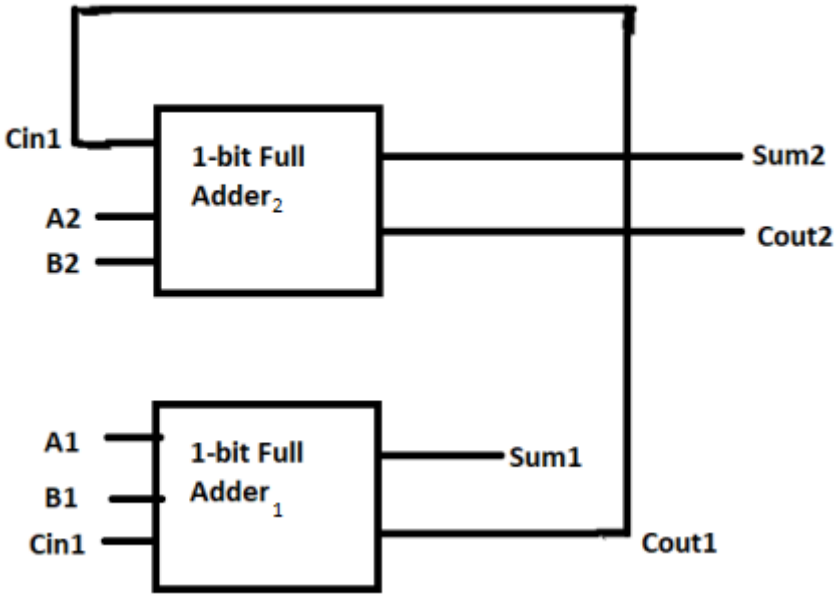
Truth Tables for 1-bit full-adder & subtractor.

By using K-Maps, we can write down the boolean equations for sum, carry-out, diff and borrow-out.

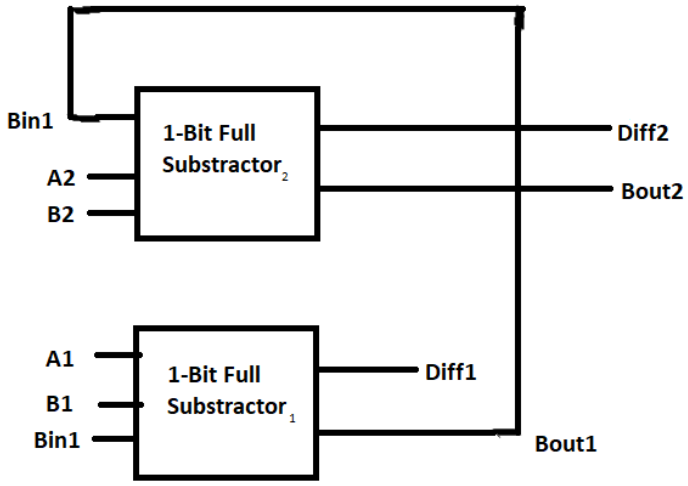
Sum = A'B'C + A'BC' + ABC + AB'C' C_{out} = BC + AB + AC

Diff = A'B'B_{in} + A'BB_{in}' + ABB_{in} + AB'B_{in}' B_{out} = BB_{in} + A'B + A'B_{in}

2. 2-Bit Full-Adder & Subtractor



Above, is the simple version of 2-bit full adder schematic, made from 1-bit full adders.



Simple version of 2-bit full subtractor schematic, made from 1-bit full subtractors.

3. SystemVerilog Modules for 1-Bit Full-Adder & Subtractor

3.1. Dataflow SystemVerilog Modules

```
3  module oneBitFullAdder( input logic a,b, c_in,
4                          output logic sum, c_out
5      );
6      assign sum =  ~a & ~b & c_in | ~a & b & ~c_in
7                                   | a & ~b & ~c_in
8                                   | a & b & c_in;
9      assign c_out = b & c_in | a & b | a & c_in;
10 endmodule
```

Dataflow SystemVerilog module for 1-Bit full-adder

```
12 module oneBitFullSubstractor( input  logic a,b, b_in,
13                               output logic dif, b_out
14 );
15     assign dif =  ~a & ~b & b_in | ~a & b & ~b_in
16                                  | a & ~b & ~b_in
17                                  | a & b & b_in;
18     assign b_out = b & b_in | ~a & b | ~a & b_in;
19 endmodule
```

Dataflow SystemVerilog module for 1-Bit full-subtractor

3.2. Structural SystemVerilog Modules

```
21  module and3( input logic a,b,c,
22                  output logic y
23      );
24      assign y = a & b & c;
25  endmodule
26
27  module and2( input logic a,b,
28                  output logic y
29      );
30      assign y = a & b;
31  endmodule
```

```
33  module or4( input a, b, c, d,
34              output y
35      );
36      assign y = a | b | c | d;
37  endmodule
38
39  module or3( input a, b, c,
40              output y
41      );
42      assign y = a | b | c;
43  endmodule
```

```
45  module inv( input logic a,
46              output logic y
47      );
48      assign y = ~a;
49  endmodule
50
```

Structural SystemVerilog modules for 2-Bit full-adder & subtractor

```
51  module fullAdder( input logic a, b, c_in,
52                    output logic sum, c_out
53      );
54      logic notA, notB, notC_in;
55      logic pro1, pro2, pro3, pro4, pro5, pro6, pro7;
56      inv inv1( a, notA);
57      inv inv2( b, notB);
58      inv inv3( c_in, notC_in);
59      and3 and3of1( notA, notB, c_in, pro1);
60      and3 and3of2( notA, b, notC_in, pro2);
61      and3 and3of3( a, notB, notC_in, pro3);
62      and3 and3of4( a, b, c_in, pro4);
63      and2 and2of1( b, c_in, pro5);
64      and2 and2of2( a, b, pro6);
65      and2 and2of3( a, c_in, pro7);
66      or4 or4of1( pro1, pro2, pro3, pro4, sum);
67      or3 or3of1( pro5, pro6, pro7, c_out);
68  endmodule
```

Structural SystemVerilog module for 1-Bit full-adder

```
70  module fullSubstractor( input logic a, b, b_in,
71                    output logic dif, b_out
72      );
73      logic notA, notB, notB_in;
74      logic pro1, pro2, pro3, pro4, pro5, pro6, pro7;
75      inv inv1( a, notA);
76      inv inv2( b, notB);
77      inv inv3( b_in, notB_in);
78      and3 and3of1( notA, notB, b_in, pro1);
79      and3 and3of2( notA, b, notB_in, pro2);
80      and3 and3of3( a, notB, notB_in, pro3);
81      and3 and3of4( a, b, b_in, pro4);
82      and2 and2of1( b, b_in, pro5);
83      and2 and2of2( notA, b, pro6);
84      and2 and2of3( notA, b_in, pro7);
85      or4 or4of1( pro1, pro2, pro3, pro4, dif);
86      or3 or3of1( pro5, pro6, pro7, b_out);
87  endmodule
```

Structural SystemVerilog module for 1-Bit full-subtractor

3.3. Testbench for 1-Bit Full-Adder & Subtractor

```
3  module testbench1();
4      logic a, b, c_in, sum, c_out;
5      fullAdder test( a, b, c_in, sum, c_out);
6      initial begin
7          a = 0; b = 0; c_in = 0; #10;
8          c_in = 1; #10;
9          b = 1; c_in = 0; #10;
10         c_in = 1; #10;
11         a = 1; b = 0; c_in = 0; #10;
12         c_in = 1; #10;
13         b = 1; c_in = 0; #10;
14         c_in = 1; #10;
15     end
16 endmodule
```

Testbench for full-adder

```
18 module testbench2();
19     logic a, b, b_in, dif, b_out;
20     fullSubs test( a, b, b_in, dif, b_out);
21     initial begin
22         a = 0; b = 0; b_in = 0; #10;
23         b_in = 1; #10;
24         b = 1; b_in = 0; #10;
25         b_in = 1; #10;
26         a = 1; b = 0; b_in = 0; #10;
27         b_in = 1; #10;
28         b = 1; b_in = 0; #10;
29         b_in = 1; #10;
30     end
31 endmodule
```

Testbench for full-subtractor

4. SystemVerilog Modules for 2-Bit Full-Adder & Subtractor

```
3  module TwoBitAdder( input logic a_1, b_1, c_in , a_2, b_2,
4                      output logic  sum_1, c_out, sum_2
5      );
6      logic mid;
7      fullAdder adder1( a_1, b_1, c_in, sum_1, mid);
8      fullAdder adder2( a_2, b_2, mid, sum_2, c_out);
9  endmodule
10
11 module TwoBitSubtractor( input logic a_1, b_1, b_in , a_2, b_2,
12                     output logic  dif_1, b_out, dif_2
13     );
14     logic mid;
15     fullAdder adder1( a_1, b_1, b_in, dif_1, mid);
16     fullAdder adder2( a_2, b_2, mid, dif_2, b_out);
17 endmodule
```

Structural SystemVerilog for 2-Bit full-adder & subtractor

```
33 module testbench3();
34     logic a_1, b_1, a_2, b_2, sum_1, sum_2, c_out;
35     logic dif_1, dif_2, b_out;
36     TwoBitAdder test1( a_1, b_1, 0, a_2, b_2, sum_1, c_out, sum_2);
37     TwoBitSubstractor test2( a_1, b_1, 0, a_2, b_2, sum_1, c_out, sum_2);
38     initial begin
39         a_1 = 0; b_1 = 0; a_2 = 0; b_2 = 0; #20; // 0000
40         b_2 = 1; #20;                           // 0001
41         a_2 = 1; b_2 = 0; #20;                  // 0010
42         b_1 = 1; a_2 = 0; #20;                  // 0100
43         a_1 = 1; b_1 = 0;  #20;                 // 1000
44         b_1 = 1; #20;                           // 1100
45         a_2 = 1; b_1 = 0; #20;                  // 1010
46         b_2 = 1; a_2 = 0; #20;                  // 1001
47         a_1 = 0; a_2 = 1; #20;                  // 0011
48         a_2 = 0; b_1 = 1; #20;                  // 0101
49         b_2 = 0; a_2 = 1; #20;                  // 0110
50         b_2 = 1; #20;                           // 0111
51         b_1 = 0; a_1 = 1; #20;                  // 1011
52         a_2 = 0; b_1 = 0; #20;                  // 1101
53         b_2 = 0; a_2 = 1; #20;                  // 1110
54         b_2 = 1; #20;                           // 1111
55     end
56 endmodule
```

Testbench for both modules, only when b_in and c_in are 0