CS 223 - Digital Design

Melike Demirci

21702346

Section 4

Lab02

02/03/2020

b) Circuit schematic for full adder using 2-input XOR, AND and OR gates

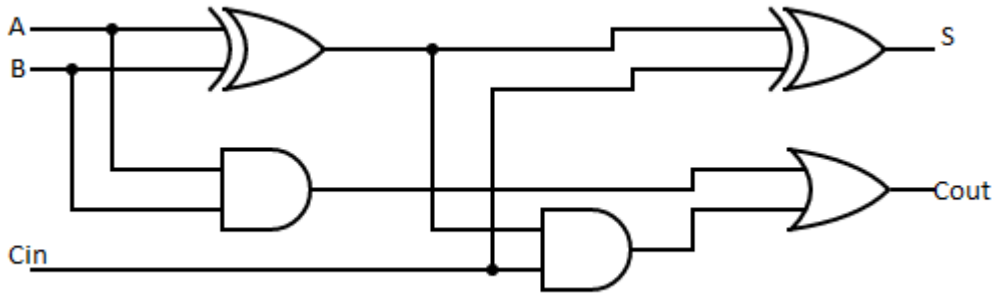

Figure 1 – Full Adder Circuit Schematic

S = (A⊕B) ⊕Cin
Cout = AB + Cin(A⊕B)

(c) Circuit schematic for full subtractor

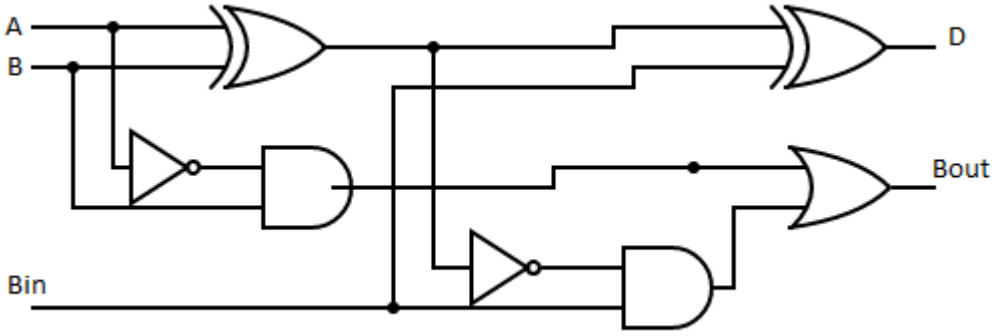

Figure 2 – Full Substructor Circuit Schematic

D = (A⊕B) ⊕Bin

Bout = ~AB + Bin ~ (A⊕B)

(d) Circuit schematic for a 2-bit adder made from two full adders

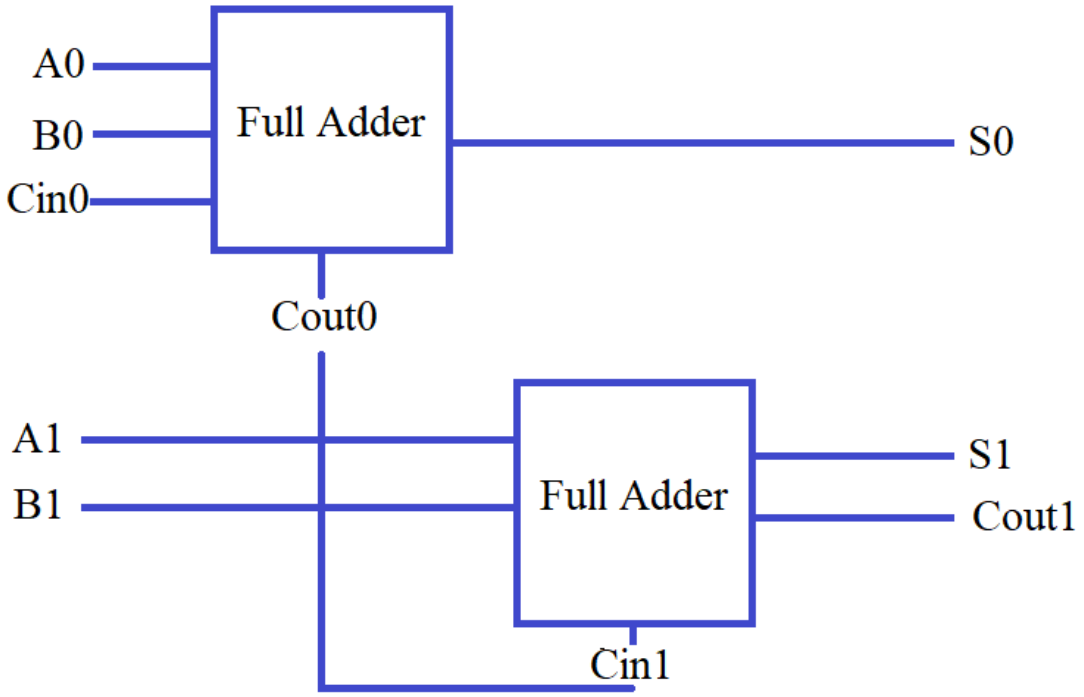

Figure 3 – 2-bit Adder Circuit Schematic

(e) Behavioral System Verilog module for the full adder and a testbench for it.

```
//Behavioral System Verilog module for a 1-bit Full Adder
module one_bit_full_adder_behavioral (input logic a, b, cin, output logic s, cout);
        assign s = a ^ b ^ cin;
        assign cout = a & b | (a ^ b) & cin;
endmodule


//Testbench
module one_bit_full_adder_behavioral_tb();
        logic a, b, cin, s, cout;
        one_bit_full_adder_behavioral fab(a, b, cin, s, cout);
        initial begin
                a = 0; b = 0; cin = 0;          #10
                cin = 1;                        #10
                b = 1; cin = 0;                 #10
                cin = 1;                        #10
                a = 1; b = 0; cin = 0;          #10
                cin = 1;                        #10
                b = 1; cin = 0;                 #10
                cin = 1;                        #10
        end
endmodule
```

(f) Structural System Verilog module for the full adder and a testbench for it.

```
//Structural System Verilog module for a 1-bit Full Adder
module one_bit_full_adder_structural (a, b, cin, s, cout);
        logic n1, n2, n3;
        xor xor_1(n1, a, b);
        xor xor_2(s, n1, cin);
        and and_1(n2, a, b);
        and and_2(n3, n1, n2);
        or or_1(cout, n3, n2);
end module
//Testbench
module one_bit_full_adder_structural_tb();
        logic a, b, cin, s, cout;
        one_bit_full_adder_structural fas(a, b, cin, s, cout);
        initial begin
                a = 0; b = 0; cin = 0;          #10
                cin = 1;                        #10
                b = 1; cin = 0;                 #10
                cin = 1;                        #10
                a = 1; b = 0; cin = 0;          #10
                cin = 1;                        #10
                b = 1; cin = 0;                 #10
                cin = 1;                        #10
        end
endmodule
```

(g) Structural System Verilog module for the full subtractor and a testbench for it.

```
//Structural System Verilog module for a 1-bit Full Substructor
module one_bit_full_substructor_structural (a, b, bin, d, bout);
        logic n1, n2, n3;
        xor xor_1(n1, a, b);
        xor xor_2(d, n1, bin);
        and and_1(n2, ~a, b);
        and and_2(n3, ~n1, bin);
        or or_1(bout, n3, n2);
end module
//Testbench
module one_bit_full_subtructor_structural_tb();
        logic a, b, bin, d, bout;
        one_bit_full_subtructor_structural fss(a, b, bin, d, bout);
initial begin
                a = 0; b = 0; bin = 0;          #10
                bin = 1;                        #10
                b = 1; bin = 0;                 #10
                bin = 1;                        #10
                a = 1; b = 0; bin = 0;          #10
                bin = 1;                        #10
                b = 1; bin = 0;                 #10
                bin = 1;                        #10
        end
endmodule
```

(h) Structural System Verilog module for the 2-bit adder and a testbench for it.

```
//Structural System Verilog module for a 2-bit adder
module two_bit_adder (input logic, a0, a1, b0, b1, cin0, output logic s0, s1, cout1);
        logic cout;
        one_bit_full_adder_structural full_adder(a0, b0, cin0, s0, cout);
        one_bit_full_adder_structural full_adder2(a1, b1, cout, s1, cout1)
endmodule
//Testbench
module two_bit_adder_tb();
        logic a0, a1, b0, b1, cin0;
        logic s0, s1, cout1;
        two_bit_adder tba(a0, a1, b0, b1, cin0, s0, s1, cout1);
        initial begin
                a0= 0; a1=0; b0=0; b1=0; cin0=0; #10; //00000 failed
                a0= 0; a1=0; b0=0; b1=0; cin0=1; #10; //00001 failed
                a0= 0; a1=0; b0=0; b1=1; cin0=0; #10; //00010 failed
                a0= 0; a1=0; b0=0; b1=1; cin0=1; #10; //00011 failed
                a0= 0; a1=0; b0=1; b1=0; cin0=0; #10; //00100 failed
                a0= 0; a1=0; b0=1; b1=0; cin0=1; #10; //00101 failed
                a0= 0; a1=0; b0=1; b1=1; cin0=0; #10; //00110 failed
                a0= 0; a1=0; b0=1; b1=1; cin0=1; #10; //00111 failed
                a0= 0; a1=1; b0=0; b1=0; cin0=0; #10; //01000 failed
                a0= 0; a1=1; b0=0; b1=0; cin0=1; #10; //01001 failed
                a0= 0; a1=1; b0=0; b1=1; cin0=0; #10; //01010 failed
                a0= 0; a1=1; b0=0; b1=1; cin0=1; #10; //01011 failed
                a0= 0; a1=1; b0=1; b1=0; cin0=0; #10; //01100 failed
                a0= 0; a1=1; b0=1; b1=0; cin0=1; #10; //01101 failed
```

```
                a0= 0; a1=1; b0=1; b1=1; cin0=0; #10; //01110 failed
                a0= 0; a1=1; b0=1; b1=1; cin0=1; #10; //01111 failed
                a0= 1; a1=0; b0=0; b1=0; cin0=0; #10; //10000 failed
                a0= 1; a1=0; b0=0; b1=0; cin0=1; #10; //10001 failed
                a0= 1; a1=0; b0=0; b1=1; cin0=0; #10; //10010 failed
                a0= 1; a1=0; b0=0; b1=1; cin0=1; #10; //10011 failed
                a0= 1; a1=0; b0=1; b1=0; cin0=0; #10; //10100 failed
                a0= 1; a1=0; b0=1; b1=0; cin0=1; #10; //10101 failed
                a0= 1; a1=0; b0=1; b1=1; cin0=0; #10; //10110 failed
                a0= 1; a1=0; b0=1; b1=1; cin0=1; #10; //10111 failed
                a0= 1; a1=1; b0=0; b1=0; cin0=0; #10; //11000 failed
                a0= 1; a1=1; b0=0; b1=0; cin0=1; #10; //11001 failed
                a0= 1; a1=1; b0=0; b1=1; cin0=0; #10; //11010 failed
                a0= 1; a1=1; b0=0; b1=1; cin0=1; #10; //11011 failed
                a0= 1; a1=1; b0=1; b1=0; cin0=0; #10; //11100 failed
                a0= 1; a1=1; b0=1; b1=0; cin0=1; #10; //11101 failed
                a0= 1; a1=1; b0=1; b1=1; cin0=0; #10; //11110 failed
                a0= 1; a1=1; b0=1; b1=1; cin0=1; #10; //11111 failed
        end
endmodule
```