CS 223 Digital Design

Bilkent University
Spring 2021/2022

Laboratory Assignment 2
Preliminary Report

Section 2

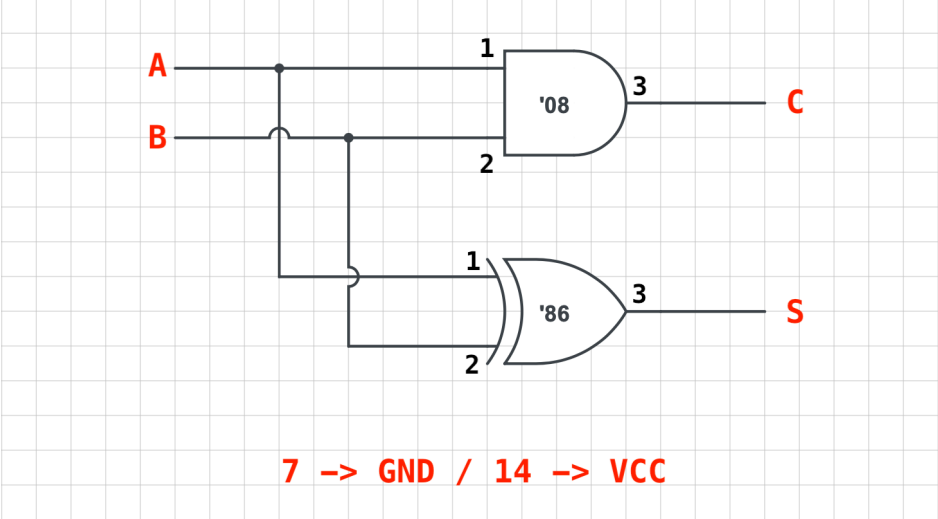
Deniz Tuna Onguner

22001788

Mon. February 28th, 2022

1 Circuit Schematics

1.1 Circuit schematic of a half adder

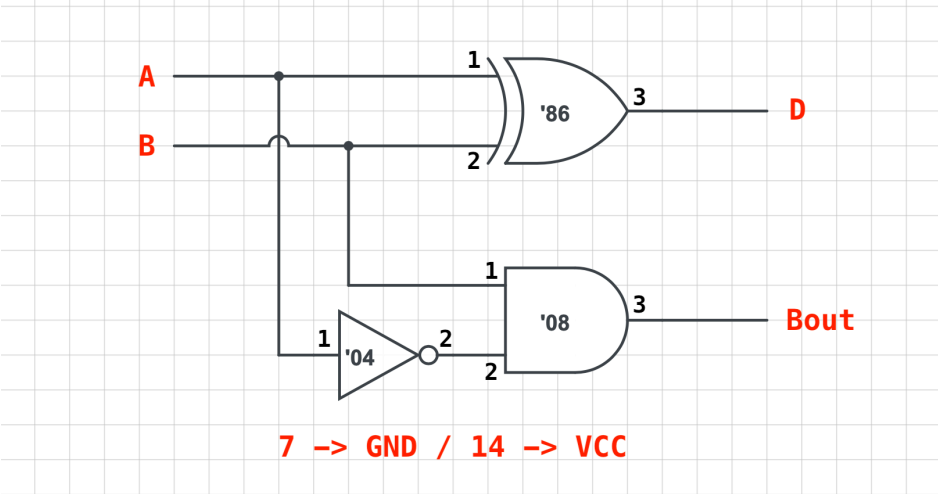


IC List:

1x 74LS08 — Two Input Quadruple AND Gate

1x 74LS86 — Two Input Quadruple XOR Gate

1.2 Circuit schematic of a half subtractor



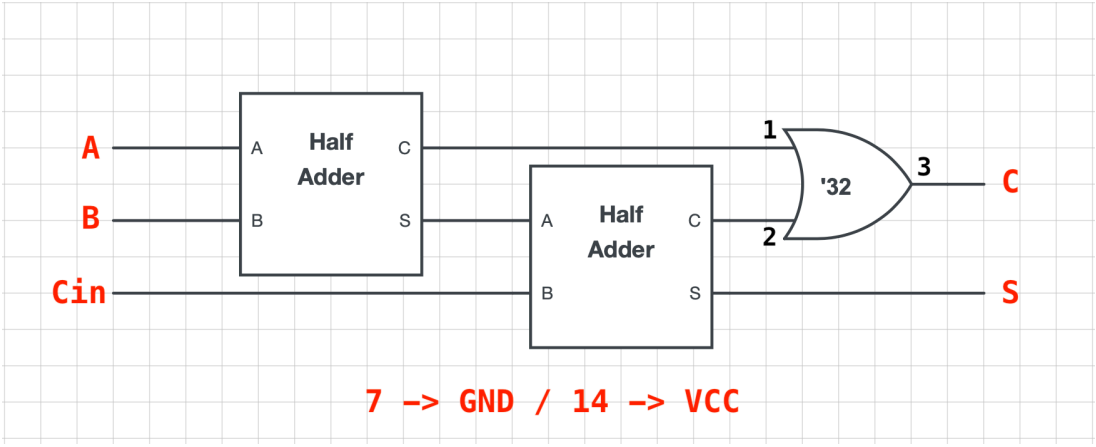
IC List:

1x 74LS08 — Two Input Quadruple AND Gate

1x 74LS86 — Two Input Quadruple XOR Gate

1x 74LS04 — Two Input Quadruple NOT Gate

1.3 Circuit schematic of a full adder by two half adders



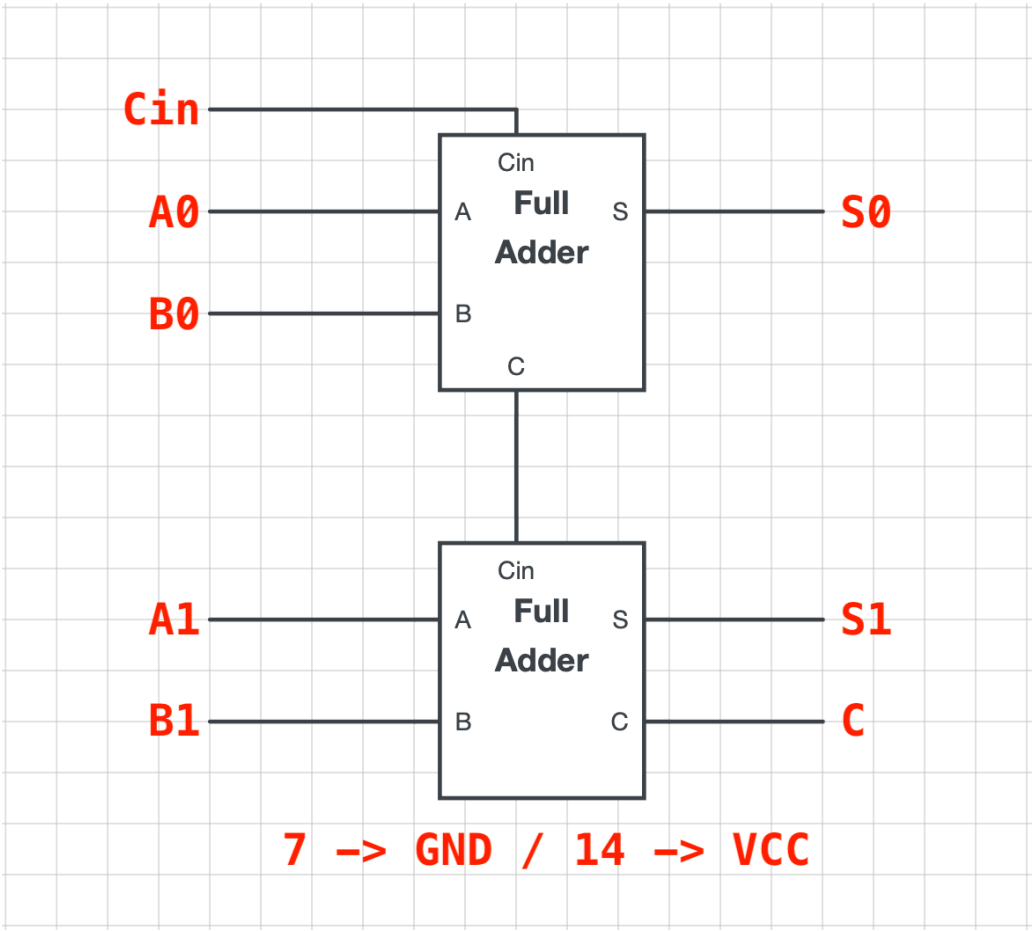
IC List:

1x 74LS08 — Two Input Quadruple AND Gate

1x 74LS86 — Two Input Quadruple XOR Gate

1x 74LS32 — Two Input Quadruple OR Gate

1.4 Circuit schematic of a 2-bit adder by two full adders



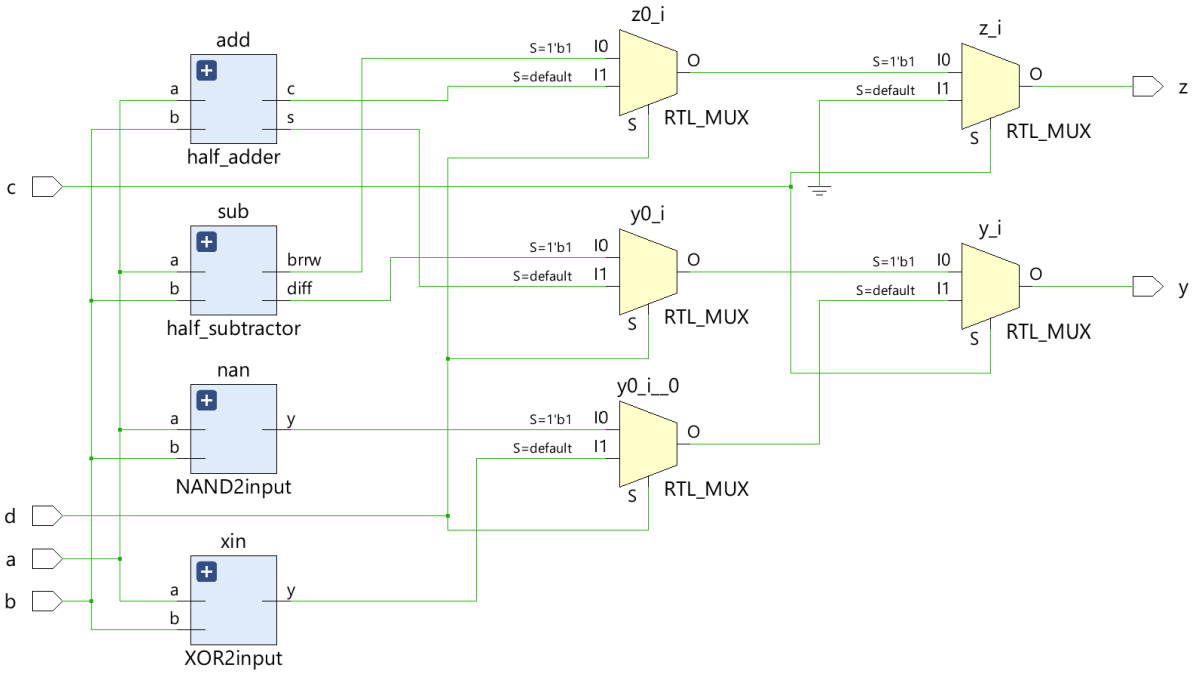
IC List:

1x 74LS08 — Two Input Quadruple AND Gate

1x 74LS86 — Two Input Quadruple XOR Gate

1x 74LS32 — Two Input Quadruple OR Gate

1.5 Circuit schematic of the lab calculator



Drew by Vivado*

IC List:

1x 74LS86 — Two Input Quadruple XOR Gate

1x 74LS08 — Two Input Quadruple AND Gate

1x 74LS04 — Two Input Quadruple NOT Gate

5x 2-1 multiplexer

*** "CircuitLab" online schematic editor is used to design and implement circuit schematics in this paper:

Online circuit simulator & schematic editor. CircuitLab. (n.d.). Retrieved February 24, 2022, from https://www.circuitlab.com/

2 System Verilog Modules

2.1 System Verilog implementation of the half adder

```
module half_adder(input logic a, b,
                  output logic s, c);
     assign s = a ^ b;
     assign c = a & b;
endmodule
```

Testbench:

```
module half_adder_TB();
     logic a, b;
     logic c, s;
     half_adder dut(a, b, s, c);
     initial begin
          a = 0; b = 0; #10;
          if (c != 0 | s != 0) $display("00 failed");
          b = 1; #10;
          if (c != 0 | s != 1) $display("01 failed");
          a = 1; #10;
          if (c != 1 | s != 0) $display("10 failed");
          b = 0; #10;
          if (c != 0 | s != 1) $display("11 failed");
     end
endmodule
```

2.2 System Verilog implementation of the half subtractor

```
module half_subtractor(input logic a, b,
                       output logic diff, brrw);
     assign diff =  a ^ b;
     assign brrw = ~a & b;
endmodule
```

Testbench:

```
module half_subtractor_TB();
     logic a, b,
     logic diff, brrw;
     half_subtractor dut(a, b, diff, brrw);
     initial begin
          a = 0; b = 0; #10;
          if (diff != 0 | brrw != 0) $display("00 failed");
          b = 1; #10;
          if (diff != 1 | brrw != 1) $display("01 failed");
          a = 1; #10;
          if (diff != 0 | brrw != 0) $display("10 failed");
          b = 0; #10;
          if (diff != 1 | brrw != 0) $display("11 failed");
     end
endmodule
```

2.3 System Verilog implementation of the full adder

```
module OR2input(input logic a, b, output logic y);
     assign y = a | b;
endmodule

module full_adder(input logic a, b, cin, output logic s, c);
     logic n0, n1, n2;
     half_adder ha1(a, b, n0, n1);
     half_adder ha2(n0, cin, s, n2);
     OR2input or2in(n2, n1, c);
endmodule
```

Testbench:

```
module full_adder_TB();
     logic a, b, cin, s, c;
     full_adder dut(a, b, cin, s, c);
     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
```

2.4 System Verilog implementation of the 2-bit adder

```
module two_bit_adder(input logic a0, a1, b0, b1, cin,
                     output logic s0, s1, c);
     logic c0;
     full_adder fa1(a0, b0, cin, s0, c0);
     full_adder fa2(a1, b1, c0, s1, c);
endmodule
```

Testbench:

```
module two_bit_adder_TB();
     logic a0, a1, b0, b1, cin;
     logic s0, s1, c;
     two_bit_adder dut(a0, a1, b0, b1, cin, s0, s1, c);
     initial begin
          a0 = 0; a1 = 0; b0 = 0; b1 = 0; cin = 0; #10;
          a0 = 1;                                  #10;
          a0 = 0; a1 = 1;                          #10;
          a0 = 1; a1 = 0; b0 = 1;                  #10;
          a0 = 0; a1 = 1; b0 = 0; b1 = 1;          #10;
                                  b1 = 0; cin = 1; #10;
          a0 = 1; a1 = 0; b0 = 1;                  #10;
                  a1 = 1;         b1 = 1;          #10;
     end
endmodule
```

2.5 System Verilog implementation of the lab calculator

```
module XOR2input(input logic a, b, output logic y);
     assign y = a ^ b;
endmodule

module NAND2input(input logic a, b, output logic y);
     assign y = ~(a & b);
endmodule

module LabCalculator(input logic c, d, a, b,
                     output logic y, z);
     logic sum, diff, cout, brrw, notin, xorin;
     half_adder add(a, b, sum, cout);
     half_subtractor sub(a, b, diff, brrw);
     XOR2input xin(a, b, xorin);
     NAND2input nan(a, b, notin);
     assign y = c ? ( d ? diff : sum ) : ( d ? notin :
     xorin);
     assign z = c ? ( d ? brrw : cout ) : 0;
endmodule
```

Testbench:

```
module LabCalculator_TB();
     logic c, d, a, b, y, z;
     LabCalculator dut(c, d, a, b, y, z);
     initial begin
     c = 0; d = 0; a = 0; b = 0; #10;
                           b = 1; #10;
                  a = 1; b = 0; #10;
                           b = 1; #10;
            d = 1; a = 0; b = 0; #10;
                           b = 1; #10;
                  a = 1; b = 0; #10;
                           b = 1; #10;
     c = 1; d = 0; a = 0; b = 0; #10;
                           b = 1; #10;
                  a = 1; b = 0; #10;
                           b = 1; #10;
            d = 1; a = 0; b = 0; #10;
                           b = 1; #10;
                  a = 1; b = 0; #10;
                           b = 1; #10;
     end
endmodule
```