CS 223 Digital Design

Section 5

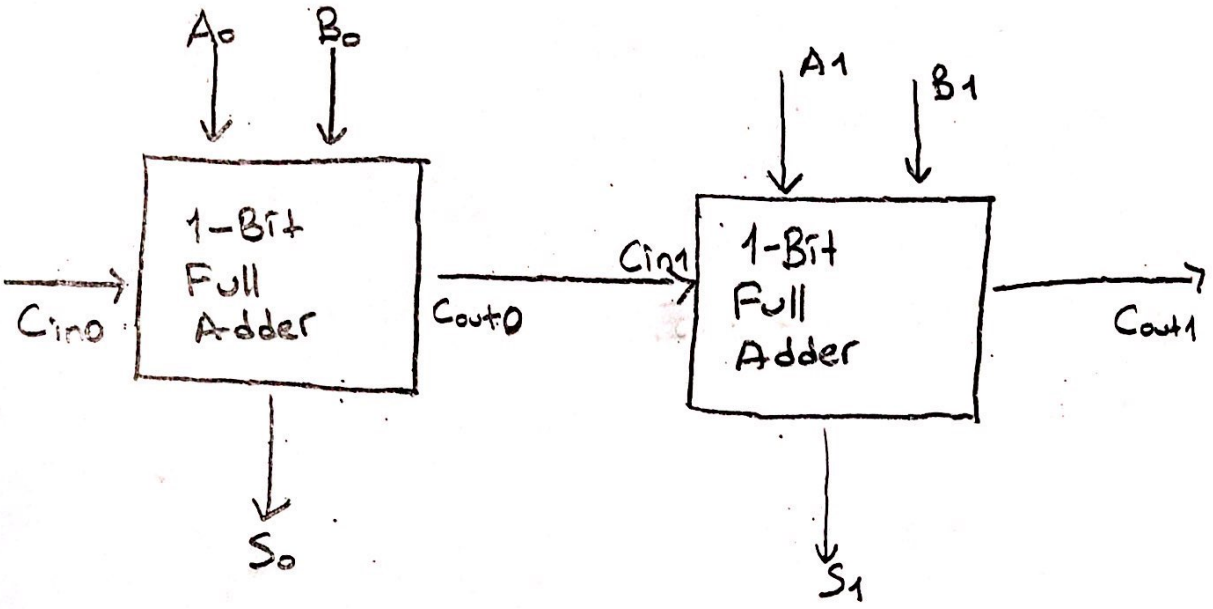
Lab 2

Name: Esad Ismail Tok

ID: 21801679

Date: 11.10.2020

~ 2-Bit Full Adder ~



IC List

One 7486 Quad 2-Input XOR Gate
One 7408 Quad 2-Input AND Gate
One 7432 Quad 2-Input OR Gate

7486	7408	7432
GND-7	GND-7	GND-7
Vcc-14	Vcc-14	Vcc-14

Behavioral SystemVerilog Module for the Full Adder and TestBench

```
`timescale 1ns / 1ps


module fullAdderBehavioral(input logic a, b, cin, output logic sum, cout);

    assign sum = (a ^ b) ^ cin;
    assign cout = (a & b) | cin & (a ^ b);

endmodule

// TestBench for the Behavioral Full Adder Module
module testbenchPartA();

    // Internal input variables fot he uut
    logic a, b, cin;

    // Internal output variables for the uut
    logic sum, cout;

    // Instanciating the unit under test
    fullAdderBehavioral uut(a, b, cin, sum, cout);

    // Applying inputs and checking the outputs
    initial begin
        a = 0; b = 0; cin = 0; #10;
        if (sum === 0 && cout === 0) $display("000 Correct");

        a = 0; b = 0; cin = 1; #10;
        if (sum === 1 && cout === 0) $display("001 Correct");

        a = 0; b = 1; cin = 0; #10;
        if (sum === 1 && cout === 0) $display("010 Correct");

        a = 0; b = 1; cin = 1; #10;
        if (sum === 0 && cout === 1) $display("011 Correct");

        a = 1; b = 0; cin = 0; #10;
        if (sum === 1 && cout === 0) $display("100 Correct");

        a = 1; b = 0; cin = 1; #10;
        if (sum === 0 && cout === 1) $display("101 Correct");

        a = 1; b = 1; cin = 0; #10;
        if (sum === 0 && cout === 1) $display("110 Correct");

        a = 1; b = 1; cin = 1; #10;
        if (sum === 1 && cout === 1) $display("111 Correct");
    end

endmodule
```

Structural SystemVerilog Module for the Full Adder and TestBench

```
`timescale 1ns / 1ps


// Top level module implementation
module fullAdderStructural(input logic a, b, cin, output logic sum, cout);

    // Internal nodes
    logic n1, n2, n3;

    // Assigning the output sum using structural design
    xor2 firstXor(a, b, n1);
    xor2 secondXor(n1, cin, sum);

    // Assigning the output cout using structural design
    and2 firstAnd(a, b, n2);
    and2 secondAnd(n1, cin, n3);
    or2 firstOr(n2, n3, cout);

endmodule

// Low level and gate module for the structural implementation of the full adder
module and2(input logic a, b, output logic y);
    assign y = a & b;
endmodule

// Low level or gate module for the structural implementation of the full adder
module or2(input logic a, b, output logic y);
    assign y = a | b;
endmodule

// Low level xor gate module for the structural implementation of the full adder
module xor2(input logic a, b, output logic y);
    assign y = a ^ b;
endmodule


// TestBench for the Structural Full Adder Module
module testbenchPartB();

    // Internal input variables fot he uut
    logic a, b, cin;

    // Internal output variables for the uut
    logic sum, cout;

    // Instanciating the unit under test
    fullAdderStructural uut(a, b, cin, sum, cout);

    // Applying inputs and checking the outputs
    initial begin
        a = 0; b = 0; cin = 0; #10;
        if (sum === 0 && cout === 0) $display("000 Correct");

        a = 0; b = 0; cin = 1; #10;
        if (sum === 1 && cout === 0) $display("001 Correct");
```

```
        a = 0; b = 1; cin = 0; #10;
        if (sum === 1 && cout === 0) $display("010 Correct");

        a = 0; b = 1; cin = 1; #10;
        if (sum === 0 && cout === 1) $display("011 Correct");

        a = 1; b = 0; cin = 0; #10;
        if (sum === 1 && cout === 0) $display("100 Correct");

        a = 1; b = 0; cin = 1; #10;
        if (sum === 0 && cout === 1) $display("101 Correct");

        a = 1; b = 1; cin = 0; #10;
        if (sum === 0 && cout === 1) $display("110 Correct");

        a = 1; b = 1; cin = 1; #10;
        if (sum === 1 && cout === 1) $display("111 Correct");
    end

endmodule
```

Structural SystemVerilog Module for the Full Subtractor and TestBench

```
`timescale 1ns / 1ps


// Top level module for the Full Subtractor
module fullSubtractorStructural(input logic a, b, bin, output logic d, bout);

    // Internal signals
    logic n1, n2, n3;

    // Assigning the output d
    halfSubtractor firstHalf(a, b, n1, n2);
    halfSubtractor secondHalf(n1, bin, d, n3);

    // Assigning the output bout
    or2 firstOr(n3, n2, bout);

endmodule

// Low level half subtractor module for the structural implementation of the full subtractor
module halfSubtractor(input logic a, b, output logic d, bout);
    // Internal signals
    logic n1, n2;

    // Assigning the d output
    xor2 firstXor(a, b, d);

    // Assigning the bout output
    inv firstInv(a, n1);
    and2 firstAnd(n1, b, bout);
endmodule

// Low level not gate module for the structural implementation of the full subtractor
module inv(input logic a, output logic y);
    assign y = ~a;
endmodule

// Low level and gate module for the structural implementation of the full subtractor
module and2(input logic a, b, output logic y);
    assign y = a & b;
endmodule

// Low level or gate module for the structural implementation of the full subtractor
module or2(input logic a, b, output logic y);
    assign y = a | b;
endmodule

// Low level xor gate module for the structural implementation of the full subtractor
module xor2(input logic a, b, output logic y);
    assign y = a ^ b;
endmodule

// TestBench module for the top level full subtractor
module testbenchPartC();

    // Assigning the interal signals
    logic a, b, bin;
```

```
    logic d, bout;

    // Instantiating the uut
    fullSubtractorStructural uut(a, b, bin, d, bout);

    // Applying inputs and checking the outputs
    initial begin

        a = 0; b = 0; bin = 0; #10;
        if (d === 0 && bout === 0) $display("000 Correct");

        a = 0; b = 0; bin = 1; #10;
        if (d === 1 && bout === 1) $display("001 Correct");

        a = 0; b = 1; bin = 0; #10;
        if (d === 1 && bout === 1) $display("010 Correct");

        a = 0; b = 1; bin = 1; #10;
        if (d === 0 && bout === 1) $display("011 Correct");

        a = 1; b = 0; bin = 0; #10;
        if (d === 1 && bout === 0) $display("100 Correct");

        a = 1; b = 0; bin = 1; #10;
        if (d === 0 && bout === 0) $display("101 Correct");

        a = 1; b = 1; bin = 0; #10;
        if (d === 0 && bout === 0) $display("110 Correct");

        a = 1; b = 1; bin = 1; #10;
        if (d === 1 && bout === 1) $display("111 Correct");

    end

endmodule
```

Structural SystemVerilog Module for the 2-Bit Adder and TestBench

```
`timescale 1ns / 1ps
`include "fullAdderBehavioral.sv" // Including the 1-bit full adders module to the file


module adderTwoBit(input logic a0, b0, a1, b1, cin, output logic s0, s1, cout);

    // Internal signals
    // n1 is the cout of the first 1-bit adder
    logic n1;

    // Assigns the first sum
    fullAdderBehavioral firstAdder(a0, b0, cin, s0, n1);

    // Assigns the second sum and the final carry out
    fullAdderBehavioral secondAdder(a1, b1, n1, s1, cout);

endmodule


// TestBench for the structural 2-bit full adder
module testbench2Bit();

    // Internal input variables fot he uut
    logic a0, b0, a1, b1, cin;

    // Internal output variables for the uut
    logic s0, s1, cout;

    // Instanciating the unit under test
    adderTwoBit uut(a0, b0, a1, b1, cin, s0, s1, cout);

    // Applying inputs and checking the outputs
    initial begin

        $monitor(a0, b0, a1, b1, cin);

        for (int i = 0; i < 32; i = i+1) begin

            {a0, b0, a1 ,b1, cin} = i; #10;
            $display("Value of s0 = ", s0);
            $display("Value of s1 = ", s1);
            $display("Value of cout = ", cout);
            $display();

        end

    end

endmodule
```