CS 223 - Digital Design

Laboratory Assignment 2

Full adder, full subtractor and 2-bit adder on FPGA

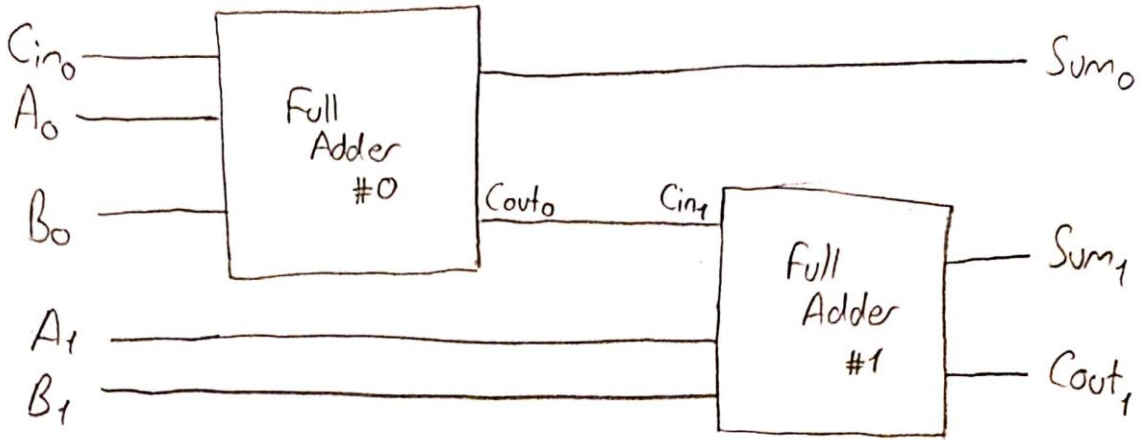
Arda İçöz

21901443

Section 02

17/10/2020

Part B: Circuit schematic for 2-bit adder



IC List:

Each Full Adder consists of:

- Two 7486 Quad 2-input XOR gates
- Two 7408 Quad 2-input AND gates
- One 7432 Quad 2-input OR gates

Thus, in 2-bit adder there are total of:

- Four 7486 Quad 2-input XOR gates
- Four 7408 Quad 2-input AND gates
- Two 7432 Quad 2-input OR gates

7486	7408	7432
GND - 7	GND - 7	GND - 7
+5V - 14	+5V - 14	+5V - 14

Part C: Behavioral SystemVerilog module for the full adder.

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

    logic p, g, e;

    assign p = a ^ b; // XOR gate
    assign g = a & b; //AND gate
    assign e = cin & p;

    assign sum = cin ^ p;
    assign cout = g | e; //OR gate which gets its second input from an AND
gate
endmodule
```

Part D: Structural SystemVerilog module for the full adder and a testbench for it.

Structural module:

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

    //full adder wires
    //p is XOR gate, g is AND gate, e is AND gate
    logic p, g, e;


    xor p_xor(p, a, b);
    and g_and(g, b, a);
    and e_and(e, p, cin);
    xor sum_xor(sum, cin, p);
    or cout_or(cout, g, e);
endmodule
```

Testbench:

```
module fullAdder_structuralTestbench();
    logic a, b, cin, sum, cout;

    //instantiate device under test
    fullAdder_structural dut(a, b, cin, sum, cout);

    //apply inputs one at a time
    //checking results
    initial begin
        a = 0; b = 0; cin = 0; #10;
        assert (cout === 0 && sum === 0) else $error("000 failed.");
        a = 0; b = 0; cin = 1; #10;
        assert (cout === 0 && sum === 1) else $error("001 failed.");
        a = 0; b = 1; cin = 0; #10;
        assert (cout === 0 && sum === 1) else $error("010 failed.");
        a = 0; b = 1; cin = 1; #10;
        assert (cout === 1 && sum === 0) else $error("011 failed.");
        a = 1; b = 0; cin = 0; #10;
        assert (cout === 0 && sum === 1) else $error("100 failed.");
        a = 1; b = 0; cin = 1; #10;
        assert (cout === 1 && sum === 0) else $error("101 failed.");
        a = 1; b = 1; cin = 0; #10;
        assert (cout === 1 && sum === 0) else $error("110 failed.");
        a = 1; b = 1; cin = 1; #10;
        assert (cout === 1 && sum === 1) else $error("111 failed.");
    end
endmodule
```

Part E: Structural SystemVerilog module for the full subtractor and a testbench for it.

Structural module:

```
module fullSubtractor_structural(
    input logic a, b, bin,
    output logic d, bo);

    //first half subtractor wires
    //p is XOR gate, g is AND gate, e is NOT gate
    logic p, g, e;

    //second half subtractor wires
    //k is XOR gate, m is AND gate, n is NOT gate
    logic k, m, n;

    //first half subtractor
    xor p_xor(p, a, b);
    not e_not(e, a);
    and g_and(g, b, e);

    //second half subtractor
    xor k_xor(d, p, bin);
    not n_not(n, p);
    and m_and(m, n, bin);

    //very last AND gate
    and last_and(bo, g, m);
endmodule
```

Testbench:

```
module fullSubtractor_structuralTestbench();
    logic a, b, bin, d, bo;

    //instantiate device under test
    fullSubtractor_structural dut(a, b, bin, d, bo);

    //apply inputs one at a time
    //checking results
    initial begin
        a = 0; b = 0; bin = 0; #10;
        assert (d === 0 && bo === 0) else $error("000 failed.");
        a = 0; b = 0; bin = 1; #10;
        assert (d === 1 && bo === 1) else $error("001 failed.");
        a = 0; b = 1; bin = 0; #10;
        assert (d === 1 && bo === 1) else $error("010 failed.");
        a = 0; b = 1; bin = 1; #10;
        assert (d === 0 && bo === 1) else $error("011 failed.");
        a = 1; b = 0; bin = 0; #10;
        assert (d === 1 && bo === 0) else $error("100 failed.");
        a = 1; b = 0; bin = 1; #10;
        assert (d === 0 && bo === 0) else $error("101 failed.");
        a = 1; b = 1; bin = 0; #10;
        assert (d === 0 && bo === 0) else $error("110 failed.");
        a = 1; b = 1; bin = 1; #10;
        assert (d === 1 && bo === 1) else $error("111 failed.");
    end
endmodule
```

Part F: Structural SystemVerilog module for the 2-bit adder using full adder module and a testbench for it.

Structural module:

```
module twobitAdder_structural(
    input logic cin, a0, b0, a1, b1,
    output logic sum0, sum1, cout1);

    //First Full Adder
    //cout0 is the output of the or gate
    logic cout0;

    fullAdder_structural fullAdder0(a0, b0, cin, sum0, cout0);
    fullAdder_structural fullAdder1(a1, b1, cout0, sum1, cout1);
endmodule
```

Testbench:

```
module twobitAdder_structuralTestbench();
    logic cin, a0, b0, a1, b1, sum0, sum1, cout1;

    //instantiate device under test
    twobitAdder_structural dut(cin, a0, b0, a1, b1, sum0, sum1, cout1);

    //apply inputs one at a time
    //checking results
    initial begin
        cin = 0; a0 = 0; b0 = 0; a1 = 0;  b1 = 0; #10;
        assert (sum0 === 0 && sum1 === 0 && cout1 === 0) else $error("0:
000, 1: 00 failed.");
        cin = 0; a0 = 1; b0 = 0; a1 = 0;  b1 = 0; #10;
        assert (sum0 === 1 && sum1 === 0 && cout1 === 0) else $error("0:
010, 1: 00 failed.");
        cin = 0; a0 = 0; b0 = 0; a1 = 1;  b1 = 0; #10;
        assert (sum0 === 0 && sum1 === 1 && cout1 === 0) else $error("0:
000, 1: 10 failed.");
        cin = 0; a0 = 1; b0 = 1; a1 = 0;  b1 = 0; #10;
        assert (sum0 === 0 && sum1 === 1 && cout1 === 0) else $error("0:
011, 1: 00 failed.");
        cin = 0; a0 = 0; b0 = 0; a1 = 1;  b1 = 1; #10;
        assert (sum0 === 0 && sum1 === 0 && cout1 === 1) else $error("0:
000, 1: 11 failed.");
        cin = 1; a0 = 0; b0 = 0; a1 = 1;  b1 = 0; #10;
        assert (sum0 === 1 && sum1 === 1 && cout1 === 0) else $error("0:
100, 1: 10 failed.");
        cin = 1; a0 = 1; b0 = 1; a1 = 0;  b1 = 0; #10;
        assert (sum0 === 1 && sum1 === 1 && cout1 === 0) else $error("0:
111, 1: 00 failed.");
        cin = 1; a0 = 1; b0 = 1; a1 = 1;  b1 = 1; #10;
        assert (sum0 === 1 && sum1 === 1 && cout1 === 1) else $error("0:
111, 1: 11 failed.");
    end
endmodule
```