CS223: Digital Design

Section 01

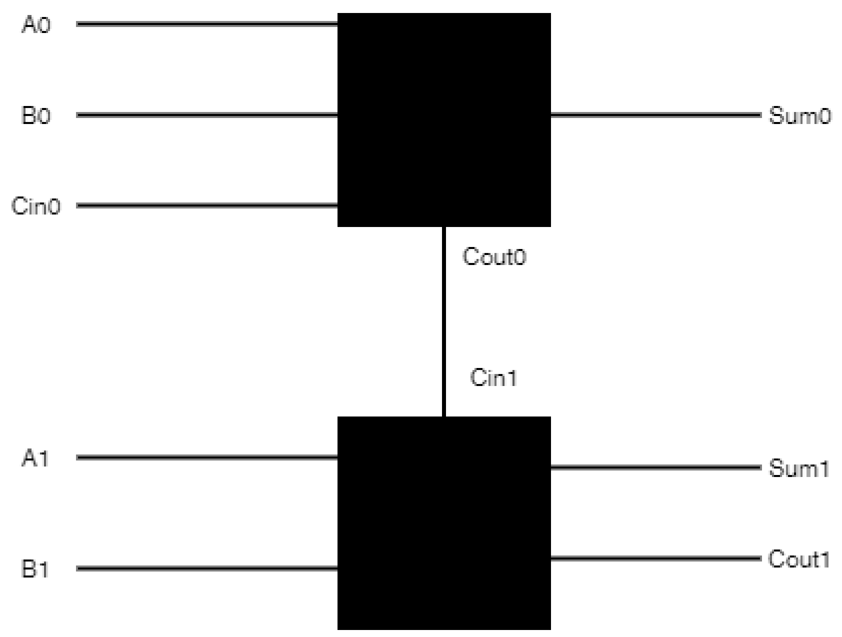
Lab 2

Efe Beydoğan

21901548

17.10.2020

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



(c) Behavioral SystemVerilog module for the full adder:

```
module behavioral_full_adder( input logic a, b, Cin, output logic sum, Cout);

   assign sum = (a ^ b) ^ Cin;

   assign Cout = ( ( a ^ b) & Cin) | ( a & b);

endmodule
```

(d) Structural SystemVerilog module for the full adder and a testbench for it:

Structural SystemVerilog module for the full adder:

```
module structural_full_adder( input logic a, b, Cin, output logic sum, Cout);

   logic n1, n2, n3; // internal nodes

   xor( n1, a, b);

   xor( sum, n1, Cin);

   and( n2, n1, Cin);

   and( n3, a, b);

   or( Cout, n2, n3);

endmodule
```

Testbench for the structural full adder:

```
module fulladder_testbench();

   logic a, b, Cin;

   logic sum, Cout;


   structural_full_adder dut( a, b, Cin, sum, Cout);


   initial begin

      a = 0; b = 0; Cin = 0; #10;

      if ( sum != 0 || Cout != 0) $display("000 failed");

      Cin = 1; #10;

      if ( sum != 1 || Cout != 0) $display("001 failed");

      b = 1; Cin = 0; #10;

      if ( sum != 1 || Cout != 0) $display("010 failed");

      Cin = 1; #10;

      if ( sum != 0 || Cout != 1) $display("011 failed");

      a = 1; b = 0; Cin = 0; #10;

      if ( sum != 1 || Cout != 0) $display("100 failed");

      Cin = 1; #10;

      if ( sum != 0 || Cout != 1) $display("101 failed");

      b = 1; Cin = 0; #10;

      if ( sum != 0 || Cout != 1) $display("110 failed");

      Cin = 1; #10;

      if ( sum != 1 || Cout != 1) $display("111 failed");

      end

   endmodule
```

(e) Structural SystemVerilog module for the full subtractor and a testbench for it:

Structural SystemVerilog module for the full subtractor:

```
module structural_full_subtractor( input logic a, b, Bin, output logic d, Bout);

   logic n1, n2, n3; // internal nodes

   xor( n1, a, b);

   xor( d, Bin, n1);

   and( n2,~a, b);

   and( n3, ~n1, Bin);

   or( Bout, n3, n2);

endmodule
```

Testbench for the structural full subtractor:

```
module fullsubtractor_testbench();

   logic a, b, Bin;

   logic d, Bout;


   structural_full_subtractor dut( a, b, Bin, d, Bout);


   initial begin

      a = 0; b = 0; Bin = 0; #10;

      if ( d != 0 || Bout != 0) $display("000 failed");

      Bin = 1; #10;

      if ( d != 1 || Bout != 1) $display("001 failed");

      b = 1; Bin = 0; #10;

      if ( d != 1 || Bout != 1) $display("010 failed");

      Bin = 1; #10;

      if ( d != 0 || Bout != 1) $display("011 failed");

      a = 1; b = 0; Bin = 0; #10;
```

```
        if ( d != 1 || Bout != 0) $display("100 failed");
        Bin = 1; #10;
        if ( d != 0 || Bout != 0) $display("101 failed");
        b = 1; Bin = 0; #10;
        if ( d != 0 || Bout != 0) $display("110 failed");
        Bin = 1; #10;
        if ( d != 1 || Bout != 1) $display("111 failed");
        end
endmodule
```

<u>**(f) Structural SystemVerilog module for the 2-bit adder and a testbench for it:**</u>

<u>**Structural SystemVerilog module for the 2-bit adder:**</u>

```
module two_bit_adder( input logic a0, b0, Cin0, a1, b1, output logic Cout1, sum1, sum0);
    logic Cout0; // internal node
    behavioral_full_adder adder0( a0, b0, Cin0, sum0, Cout0);
    behavioral_full_adder adder1( a1, b1, Cout0, sum1, Cout1);
endmodule
```

<u>**Testbench for the structural 2-bit adder:**</u>

```
module twobitadder_testbench();
    logic a0, b0, Cin0, a1, b1;
    logic sum0, sum1, Cout1;

    two_bit_adder dut( a0, b0, Cin0, a1, b1, Cout1, sum1, sum0);

    initial begin
        a0 = 0; b0 = 0; Cin0 = 0; a1 = 0; b1 = 0; #10;
```

```
if ( sum0 != 0 || sum1 != 0 || Cout1 !=0) $display("00000 failed");

b0 = 1; #10;

if ( sum0 != 1 || sum1 != 0 || Cout1 !=0) $display("01000 failed");

b0 = 0; b1 = 1; #10;

if ( sum0 != 0 || sum1 != 1 || Cout1 !=0) $display("00001 failed");

b0 = 1; #10;

if ( sum0 != 1 || sum1 != 1 || Cout1 !=0) $display("01001 failed");

a0 = 1; b0 = 0; b1 = 0; #10;

if ( sum0 != 1 || sum1 != 0 || Cout1 !=0) $display("10000 failed");

b0 = 1; #10;

if ( sum0 != 0 || sum1 != 1 || Cout1 !=0) $display("11000 failed");

b0 = 0; b1 = 1; #10;

if ( sum0 != 1 || sum1 != 1 || Cout1 !=0) $display("10001 failed");

b0 = 1; #10;

if ( sum0 != 0 || sum1 != 0 || Cout1 !=1) $display("11001 failed");

a0 = 0; a1 = 1; b0 = 0; b1 = 0; #10;

if ( sum0 != 0 || sum1 != 1 || Cout1 !=0) $display("00010 failed");

b0 = 1; #10;

if ( sum0 != 1 || sum1 != 1 || Cout1 !=0) $display("01010 failed");

b0 = 0; b1 = 1; #10;

if ( sum0 != 0 || sum1 != 0 || Cout1 !=1) $display("00011 failed");

b0 = 1; #10;

if ( sum0 != 1 || sum1 != 0 || Cout1 !=1) $display("01011 failed");

a0 = 1; b0 = 0; b1 = 0; #10;

if ( sum0 != 1 || sum1 != 1 || Cout1 !=0) $display("10010 failed");

b0 = 1; #10;

if ( sum0 != 0 || sum1 != 0 || Cout1 !=1) $display("11010 failed");
```

```
        b0 = 0; b1 = 1; #10;

        if ( sum0 != 1 || sum1 != 0 || Cout1 !=1) $display("10011 failed");

        b0 = 1; #10;

        if ( sum0 != 0 || sum1 != 1 || Cout1 !=1) $display("11011 failed");

        end

endmodule
```