ECE 426/516 Implementation of VLSI Systems with HDL
Lab 4 Verilog for 4-bit subtractor design

We have learned ripple carry adder and carry lookahead adder. A ripple-carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. It is called a ripple carry adder, because each carry bit gets rippled into the next stage. A carry-lookahead adder is a fast adder, which improves speed by reducing the amount of time required to determine carry bits.

In this lab, we expand the function of 4-bit adder design in lab3. Subtraction function is added to this module, by adding an input pin (dir). Thus, either addition or subtraction of two numbers is executed based on a control signal. Assume that the numbers are 2's complement notation.

module four_bit_rca (A, B, Cin, S, m, Cout);

If m = '0', the module is configured as a four-bit ripple carry adder.
If m = '1', the module is configured as a four-bit ripple subtractor.

Demo

You should demo the following aspects of your design to TA.

1. Verilog code for 4-bit adder/subtractor.
2. Simulation waveforms demonstrating correct functionality of these above designs.

Lab Guide

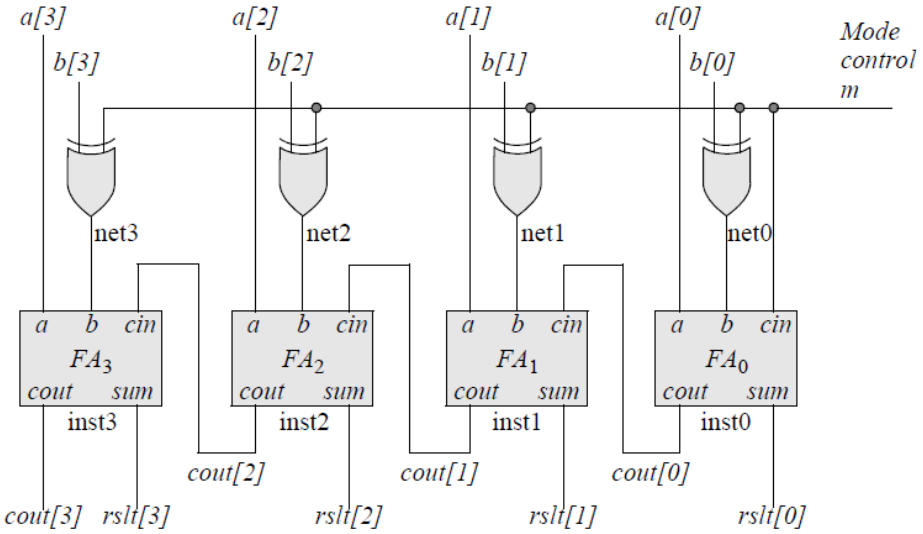

Figure Logic diagram for a 4-bit adder/subtractor

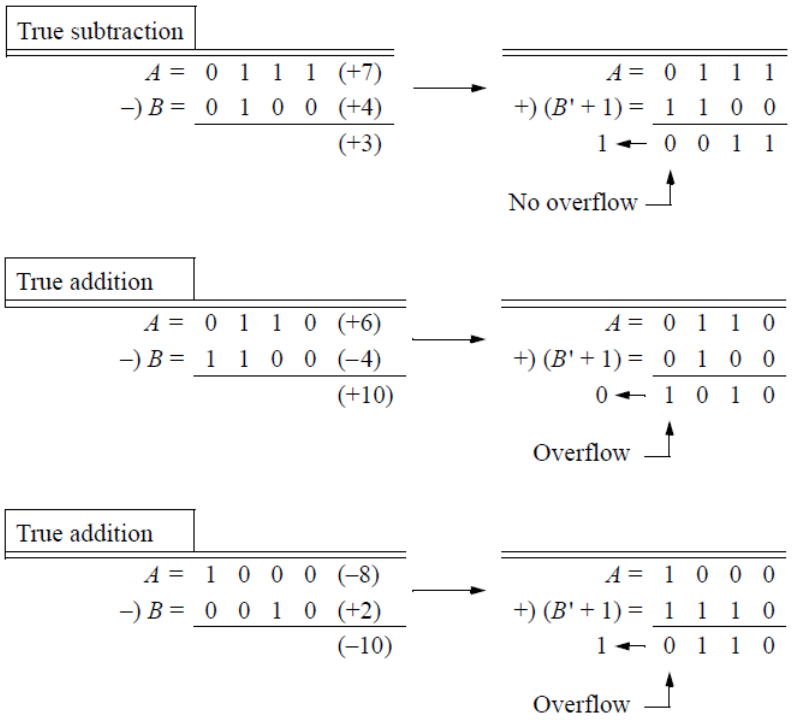

Figure Examples of fixed-point subtraction

Full adder to be instantiated into a structural module to implement a 4-bit adder/subtractor:

```
//dataflow full adder
module full_adder (a, b, cin, sum, cout);

//list all inputs and outputs
input a, b, cin;
output sum, cout;
```

```
//define wires
wire a, b, cin;
wire sum, cout;

//continuous assignment
assign sum = (a ^ b) ^ cin;
assign cout = cin & (a ^ b) | (a & b);
endmodule
```

Structural module for a 4-bit adder/subtractor:

```
`include "full_adder.v"

//structural module for an adder/subtractor
module adder_subtr_struc (a, b, m, rslt, cout, ovfl);
input [3:0] a, b;
input m;
output [3:0] rslt, cout;
output ovfl;

//define internal nets
wire net0, net1, net2, net3;
//define overflow
xor (ovfl, cout[3], cout[2]);
//instantiate the xor and the full adder for FA0
xor (net0, b[0], m);
full_adder inst0 (
.a(a[0]),
.b(net0),
.cin(m),
.sum(rslt[0]),
.cout(cout[0])
);

//instantiate the xor and the full adder for FA1
xor (net1, b[1], m);
full_adder inst1 (
.a(a[1]),
.b(net1),
.cin(cout[0]),
.sum(rslt[1]),
.cout(cout[1])
);

//instantiate the xor and the full adder for FA2
xor (net2, b[2], m);
full_adder inst2 (
.a(a[2]),
.b(net2),
.cin(cout[1]),
.sum(rslt[2]),
.cout(cout[2])
);

//instantiate the xor and the full adder for FA3
```

```
xor (net3, b[3], m);
full_adder inst3 (
.a(a[3]),
.b(net3),
.cin(cout[2]),
.sum(rslt[3]),
.cout(cout[3])
);
endmodule
```

Test bench:

```
`include "adder_subtr_struc.v"

// test bench for structural adder/subtractor

module testbench;

reg [3:0] a, b;
reg m;
wire [3:0] rslt, cout;
wire ovfl;

initial //display variables

$monitor ($time, "ns a=%b, b=%b, m=%b, rslt=%b, cout[3]=%b, cout[2]=%b, ovfl=%b",
a, b, m, rslt, cout[3], cout[2], ovfl);

initial //apply input vectors

begin

//addition
#0  a = 4'b0000; b = 4'b0001; m = 1'b0;
#10 a = 4'b0010; b = 4'b0101; m = 1'b0;
#10 a = 4'b0110; b = 4'b0001; m = 1'b0;
#10 a = 4'b0101; b = 4'b0001; m = 1'b0;

//subtraction
#10 a = 4'b0111; b = 4'b0101; m = 1'b1;
#10 a = 4'b0101; b = 4'b0100; m = 1'b1;
#10 a = 4'b0110; b = 4'b0011; m = 1'b1;
#10 a = 4'b0110; b = 4'b0010; m = 1'b1;

//overflow
#10 a = 4'b0111; b = 4'b0101; m = 1'b0;
#10 a = 4'b1000; b = 4'b1011; m = 1'b0;
#10 a = 4'b0110; b = 4'b1100; m = 1'b1;
#10 a = 4'b1000; b = 4'b0010; m = 1'b1;

#10 $stop;

end

adder_subtr_struc inst1 ( //instantiate the module
.a(a),
.b(b),
```

```
.m(m),
.rslt(rslt),
.cout(cout),
.ovfl(ovfl)
);

endmodule
```

Test bench Output:

```
Compiler version T-2022.06-SP1; Runtime version T-2022.06-SP1;  Feb  9 12:21 2023
                  0ns a=0000, b=0001, m=0, rslt=0001, cout[3]=0, cout[2]=0, ovfl=0
                 10ns a=0010, b=0101, m=0, rslt=0111, cout[3]=0, cout[2]=0, ovfl=0
                 20ns a=0110, b=0001, m=0, rslt=0111, cout[3]=0, cout[2]=0, ovfl=0
                 30ns a=0101, b=0001, m=0, rslt=0110, cout[3]=0, cout[2]=0, ovfl=0
                 40ns a=0111, b=0101, m=1, rslt=0010, cout[3]=1, cout[2]=1, ovfl=0
                 50ns a=0101, b=0100, m=1, rslt=0001, cout[3]=1, cout[2]=1, ovfl=0
                 60ns a=0110, b=0011, m=1, rslt=0011, cout[3]=1, cout[2]=1, ovfl=0
                 70ns a=0110, b=0010, m=1, rslt=0100, cout[3]=1, cout[2]=1, ovfl=0
                 80ns a=0111, b=0101, m=0, rslt=1100, cout[3]=0, cout[2]=1, ovfl=1
                 90ns a=1000, b=1011, m=0, rslt=0011, cout[3]=1, cout[2]=0, ovfl=1
                100ns a=0110, b=1100, m=1, rslt=1010, cout[3]=0, cout[2]=1, ovfl=1
                110ns a=1000, b=0010, m=1, rslt=0110, cout[3]=1, cout[2]=0, ovfl=1
$stop at time 120 Scope: testbench File: testbench.v Line: 39
ucli% exit
           V C S   S i m u l a t i o n   R e p o r t
Time: 120
CPU Time:      0.850 seconds;       Data structure size:   0.0Mb
```