Alexia Perez

ECEN 454 – 503

# Lab 1: Introduction to Cadence Schematic Capture & Simulation

September 12th, 2021

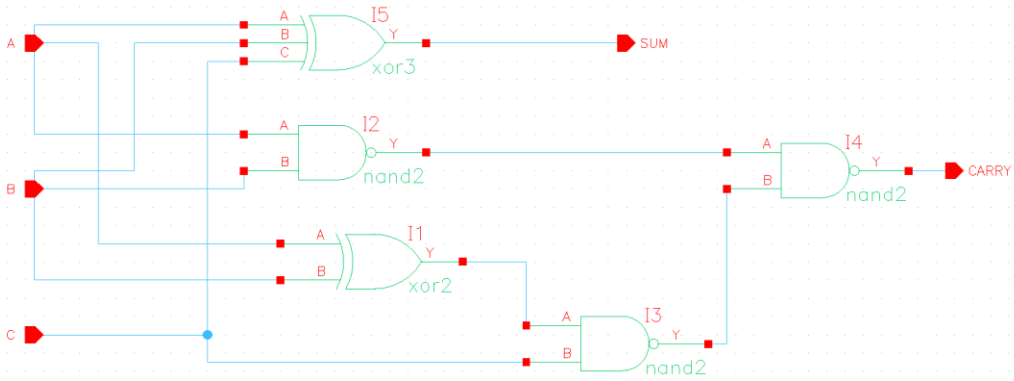
**Purpose:** This lab served to introduce me to the Cadence Virtuoso software for logic design schematic capture and simulation. A full adder, 4-bit adder, and 4-bit register will each were created, then wired together to create an 8-bit pipelined adder.

## Procedure:

1) Create a full adder schematic, and verify the design has no errors. Create a symbol for the full adder. Use this schematic to generate a testbench, and test every possible input combination on the design. Verify the simulation works as expected.
2) Create a new schematic for the 4-bit adder, and connect 4 of the previously designed full adder to make a carry-ripple adder. Create a symbol for the 4-bit adder, use the schematic to generate a testbench, and test a subset of inputs. Verify the simulation works as expected.
3) Create a new schematic for the 4-bit register. Use 4 flip-flops with clock and _clock, and allow 4 inputs and 4 outputs. Create a symbol for the 4-bit register, use the schematic to generate a testbench, and test a subset of inputs. Verify the simulation works as expected.
4) Finally, create a new schematic for the 8-bit pipelined adder. 2 4-bit adders, 3 4-bit registers, and one flip flop will be used. Wire the components as shown in Lab 0 Figure 1. Use the schematic to generate a testbench, test a subset of inputs, and verify the simulation works as expected.

## Results:

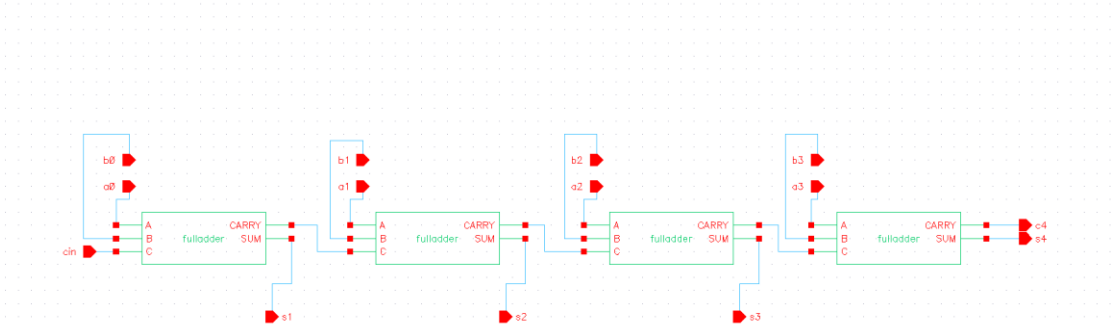
1) The full adder was created using 3 NAND gates and 2 XOR gates. Once wired up, each possible input was tested and the output was checked. The schematic, symbol, test code, and test results are displayed below.

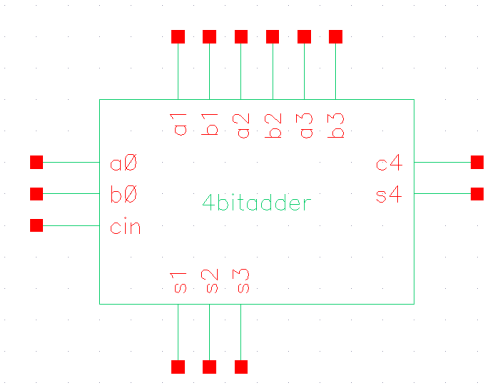



```
// Verilog stimulus file.
// Please do not create a module in this file.


// Default verilog stimulus.

initial
begin

   A = 1'b0;
   B = 1'b0;
   C = 1'b0; //ABC = 000

   #50 C = 1'b1; //ABC = 001

   #50 B = 1'b1;
       C = 1'b0; //ABC = 010

   #50 C = 1'b1; //ABC = 011

   #50 A = 1'b1;
       B = 1'b0;
       C = 1'b0; //ABC = 100

   #50 C = 1'b1; //ABC = 101

   #50 B = 1'b1;
       C = 1'b0; //ABC = 110


   #50 C = 1'b1; //ABC = 111

end

initial
$monitor($time, " A=%b, B=%b, C=%b, SUM=%b, CARRY=%b", A, B, C, SUM, CARRY);
```


```
Relinquished control to SimVision...
ncsim>
ncsim> source /opt/coe/cadence/INCISIVE152/tools/inca/files/ncsimrc
ncsim> database -open shmWave -shm -default -into shm.db
Created default SHM database shmWave
ncsim> probe -create -shm test  -all   -depth 1
Created probe 1
ncsim> run
                   0 A=0, B=0, C=0, SUM=0, CARRY=0
                  50 A=0, B=0, C=1, SUM=1, CARRY=0
                 100 A=0, B=1, C=0, SUM=1, CARRY=0
                 150 A=0, B=1, C=1, SUM=0, CARRY=1
                 200 A=1, B=0, C=0, SUM=1, CARRY=0
                 250 A=1, B=0, C=1, SUM=0, CARRY=1
                 300 A=1, B=1, C=0, SUM=0, CARRY=1
                 350 A=1, B=1, C=1, SUM=1, CARRY=1
ncsim> ^C
ncsim> exit
TOOL:   ncxlmode        15.20-s078: Exiting on Sep 10, 2021 at 07:07:57 CDT  (total: 00:03:30)
```

2) The 4-bit adder was created with 4 full adders wired together. Each adder was given a certain bit from A and B, and the Cin was wired to the Cout of the previous adder, excluding the least and most significant bits. Once completed, the circuit was given a subset of possible inputs, as designated by the lab manual, and those inputs were tested. The schematic, symbol, test code, and test results are displayed below.

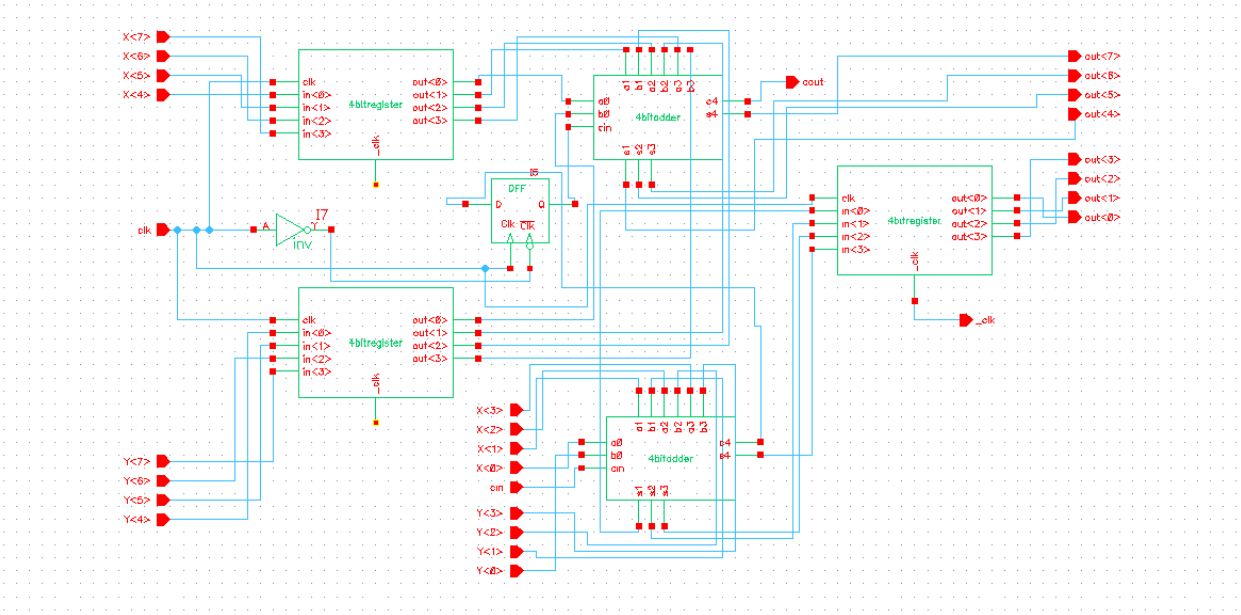



```
initial
begin

   // Initialize pins (A:0000, B:0000, cin:0)
   a0 = 1'b0;
   a1 = 1'b0;
   a2 = 1'b0;
   a3 = 1'b0;
   b0 = 1'b0;
   b1 = 1'b0;
   b2 = 1'b0;
   b3 = 1'b0;
   cin = 1'b0;


   // (Mandatory) Test 1 (A:1111, B:1111)
   #50
   a0 = 1'b1;
   a1 = 1'b1;
   a2 = 1'b1;
   a3 = 1'b1;
   b0 = 1'b1;
   b1 = 1'b1;
   b2 = 1'b1;
   b3 = 1'b1;

  // (Mandatory) Test 2 (A:1010, B:1010, cin:1)
   #50
   a0 = 1'b1;
   a1 = 1'b0;
   a2 = 1'b1;
   a3 = 1'b0;
   b0 = 1'b1;
   b1 = 1'b0;
   b2 = 1'b1;
   b3 = 1'b0;
   cin = 1'b1;


  // (Mandatory) Test 3 (A:1010, B:1010, cin:1)
   #50
   a0 = 1'b0;
   a1 = 1'b1;
   a2 = 1'b0;
   a3 = 1'b1;
   b0 = 1'b0;
   b1 = 1'b1;
   b2 = 1'b0;
   b3 = 1'b1;
   cin = 1'b1;
```

```
----------------------------------
Relinquished control to SimVision...
ncsim>
ncsim> source /opt/coe/cadence/INCISIVE152/tools/inca/files/ncsimrc
ncsim> database -open shmWave -shm -default -into shm.db
Created default SHM database shmWave
ncsim> probe -create -shm test  -all   -depth 1
Created probe 1
ncsim> run
                  0a0=0, a1=0, a2=0, a3=0, b0=0, b1=0, b2=0, b3=0, cin=0, s1=0, s2=0, s3=0, s4=0, c4=0
                 50a0=1, a1=1, a2=1, a3=1, b0=1, b1=1, b2=1, b3=1, cin=0, s1=0, s2=1, s3=1, s4=1, c4=1
                150a0=0, a1=0, a2=0, a3=0, b0=1, b1=1, b2=1, b3=1, cin=0, s1=1, s2=1, s3=1, s4=1, c4=0
                300a0=1, a1=1, a2=1, a3=1, b0=0, b1=0, b2=0, b3=0, cin=0, s1=1, s2=1, s3=1, s4=1, c4=0
                500a0=1, a1=0, a2=0, a3=0, b0=0, b1=0, b2=0, b3=1, cin=0, s1=1, s2=0, s3=0, s4=1, c4=0
                750a0=0, a1=1, a2=0, a3=0, b0=0, b1=0, b2=1, b3=0, cin=0, s1=0, s2=1, s3=1, s4=0, c4=0
               1050a0=0, a1=1, a2=1, a3=1, b0=0, b1=1, b2=1, b3=0, cin=0, s1=0, s2=0, s3=1, s4=0, c4=1
ncsim> ^C
ncsim> exit
TOOL:   ncxlmode        15.20-s078: Exiting on Sep 11, 2021 at 02:24:32 CDT  (total: 00:11:11)
```

3) For the final task, an 8-bit pipelined adder was created using two of the 4-bit adders, three of the 4-bit registers, and one D Flip Flop. The 4 least significant bits were fed straight to a 4-bit adder to obtain the cout (carry out), which was fed to the second 4-bit adder through the D Flip Flop, and the sum was fed to a 4-bit register. The 4 most significant bits of A and B were each fed to a register, then fed to a 4-bit adder along with cout from the previous adder, and that sum was combined with the sum of the first adder to obtain the final result. As designated in the lab manual, a certain subset of possible inputs was tested on this circuit. The schematic, symbol, test code, and test results are displayed below.




```
initial
begin


  cin = 1'b0;
   X[0:7] = 8'b00000000;
   Y[0:7] = 8'b00000000;
   clk = 1'b0;


   //begin test cases
   //test 1
   #20 X[0:7] = 8'b01111110; Y[0:7] = 8'b11100111;
   // test 2
   #20 cin = 1'b1; X[0:7] = 8'b11111111; Y[0:7] = 8'b00000000;
   // test 3
   #20 X[0:7] = 8'b10101010; Y[0:7] = 8'b01010101; cin = 1'b0;
   // test 4
   #20 cin = 1'b1;
   // test 5
   #20 cin = 1'b0; X[0:7] = 8'b11001100; Y[0:7] = 8'b00110011;
   //test 6
   #20 cin = 1'b1;
   #50;
   $finish;


end


always
    #10 clk = !clk;


initial
$monitor ($time," X[0:7]=%b, Y[0:7]=%b, clk=%b, _clk=%b, cin=%b, out[0:7]=%b, cout=%b", X[0:7],
```

```
Created probe 1
ncsim> run
                  0 X[0:7]=00000000, Y[0:7]=00000000, clk=0, _clk=1, cin=0, ou
t[0:7]=xxxxxxxx, cout=x
                 10 X[0:7]=00000000, Y[0:7]=00000000, clk=1, _clk=0, cin=0, ou
t[0:7]=00000000, cout=0
                 20 X[0:7]=01111110, Y[0:7]=11100111, clk=0, _clk=1, cin=0, ou
t[0:7]=00000000, cout=0
                 30 X[0:7]=01111110, Y[0:7]=11100111, clk=1, _clk=0, cin=0, ou
t[0:7]=10100110, cout=1
                 40 X[0:7]=11111111, Y[0:7]=00000000, clk=0, _clk=1, cin=1, ou
t[0:7]=10100110, cout=1
                 50 X[0:7]=11111111, Y[0:7]=00000000, clk=1, _clk=0, cin=1, ou
t[0:7]=00000000, cout=1
                 60 X[0:7]=10101010, Y[0:7]=01010101, clk=0, _clk=1, cin=0, ou
t[0:7]=00000000, cout=1
                 70 X[0:7]=10101010, Y[0:7]=01010101, clk=1, _clk=0, cin=0, ou
t[0:7]=11111000, cout=1
                 80 X[0:7]=10101010, Y[0:7]=01010101, clk=0, _clk=1, cin=1, ou
t[0:7]=11111000, cout=1
                 90 X[0:7]=10101010, Y[0:7]=01010101, clk=1, _clk=0, cin=1, ou
t[0:7]=00000100, cout=1
                100 X[0:7]=11001100, Y[0:7]=00110011, clk=0, _clk=1, cin=0, ou
t[0:7]=00000100, cout=1
                110 X[0:7]=11001100, Y[0:7]=00110011, clk=1, _clk=0, cin=0, ou
t[0:7]=11111011, cout=0
                120 X[0:7]=11001100, Y[0:7]=00110011, clk=0, _clk=1, cin=1, ou
t[0:7]=11111011, cout=0
                130 X[0:7]=11001100, Y[0:7]=00110011, clk=1, _clk=0, cin=1, ou
t[0:7]=00000111, cout=0
                140 X[0:7]=11001100, Y[0:7]=00110011, clk=0, _clk=1, cin=1, ou
t[0:7]=00000111, cout=0
                150 X[0:7]=11001100, Y[0:7]=00110011, clk=1, _clk=0, cin=1, ou
t[0:7]=00000111, cout=0
                160 X[0:7]=11001100, Y[0:7]=00110011, clk=0, _clk=1, cin=1, ou
t[0:7]=00000111, cout=0
Simulation complete via $finish(1) at time 170 NS + 0
./testfixture.verilog:45    $finish;
ncsim> ^C
ncsim> exit
TOOL:   ncxlmode        15.20-s077: Exiting on Sep 12, 2021 at 23:14:34 CDT  (t
```

## Conclusion:

Similar to Multisim or Spice, the Cadence software can be used to create hardware schematics symbols out of schematics to be used in a larger circuit. The cadence software also allows the user to create testbench files to test various inputs to the schematic and check the corresponding outputs.