Bert Zhao
EE 271

An Introduction to Verilog and Digital Components

Abstract: The purpose of this lab was to introduce the software and hardware we will be using throughout EE 271. We introduce the Quartus Prime Lite programming software as well as the FPGA development board. We also introduce the four integrated circuit chips for building basic logic circuit on the breadboard. Important topics covered in this lab include using the physical chips to create a logic circuit, creating a digital model of our physical circuit in Quartus Prime, and exporting a digital design to the DE1-SoC development board for testing.

Introduction:

This introductory lab explores the addition through circuit logic. The full adder circuit is the fundamental method used to add binary numbers. The first physical circuit of the lab is a one-bit adder which takes two one-bit inputs and a carry to produce a sum and a carry out if needed. The second circuit of this lab is an extended four bit adder which sums two four bit binary numbers producing potential five bit binary sum with carry if needed.

Materials:

- Altera Terasic DE1-SoC Board with power and USB cables
- Breadboard
- Jumper wires
- TTL Logic Chips
  - SN74LS86AL - XOR gate chip
  - SN7404AN - Inverter chip
  - SN7402N - NOR gate chip
  - SN7400N - NAND gate chip
- PC with Altera Quartus Prime Lite installed

Procedure:

Part 1: Wiring digital components and LEDs on a breadboard

The first task of the lab involved building a physical circuit of a single full adder cell. The input was set by three switches representing the A, B, and cin. The outputs were two leds for the sum and cout. We construct this circuit with only the parts available in our lab kit which means using only XOR and NAND.

$$\text{Sum} = ((AB)'[(A\oplus B)(C)]')'$$

Equation 1: Sum

$$\text{Cout} = A\oplus B\oplus C$$

Equation 2: Carry Out

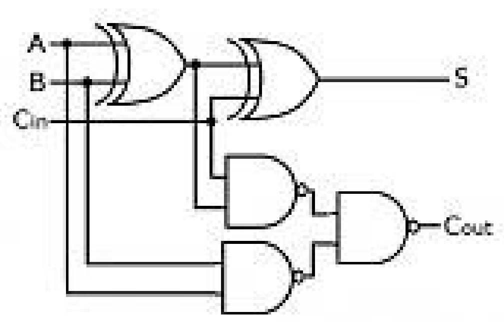


Figure 1: Schematic Diagram

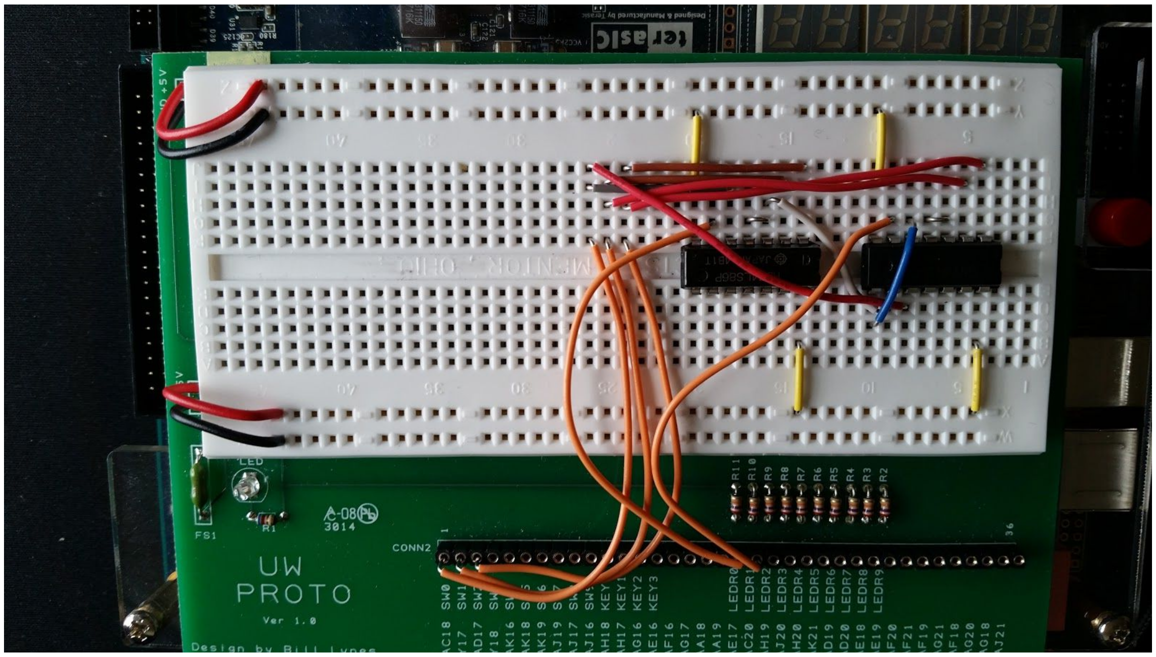


Figure 2: Breadboard Picture

Truth Table:

A	**B**	**Cin**	**Sum**	**Cout**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

Part 2: Implementation of the full adder on Quartus Prime
The second part of this lab involved a brief introduction of the Quartus Prime software. This included properly setting up a Quartus project, as well as the proper file structure necessary for compiling code to prepare for simulations. We specifically focused on modeling our previous adder cell in Quartus and then testing the circuit outputs using ModelSim.

Part 3: Design a 4-bits adder
The final part of this lab involved extending our full adder to a four-bit adder. This involved using the cout of the previous full adder cell into the cin of the next. This is a design in which the size of the full adder array can be extended indefinitely. The outputs were tested using ModelSim and then exported to the DE1-SoC development board. We could then manipulate the circuit by mapping SW0 to cin, SW1-4

to the digits of A, and SW5-9 to the digits of B. The LEDs 0-4 were also mapped to represent the sum and cout. The six hex displays were also programmed to display the word “adding.”

Results:

The breadboard building process presented a challenge because the lab kits do not include any AND and OR gates. This meant that we could not use the conventional implementations of a full adder, and instead used NAND gates to build the cout component of the full adder. For the final two parts of this lab, we tested the outputs of our circuits using ModelSim.

ModelSim Screenshots:

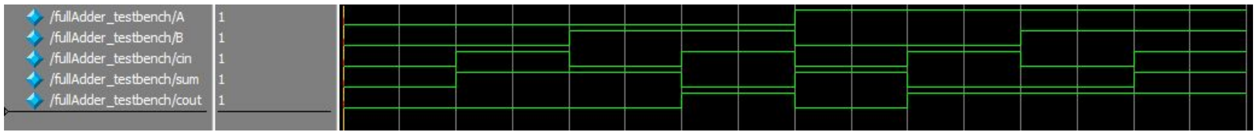

/fullAdder_testbench/A 1
/fullAdder_testbench/B 1
/fullAdder_testbench/cin 1
/fullAdder_testbench/sum 1
/fullAdder_testbench/cout 1

Figure 3: Full Adder ModelSim Screenshot

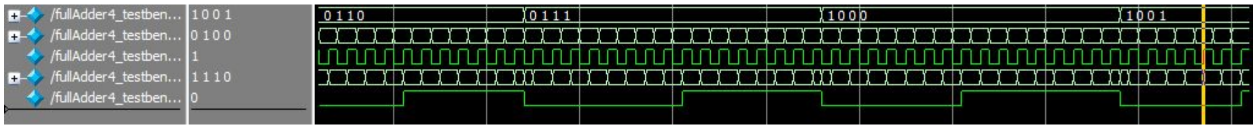
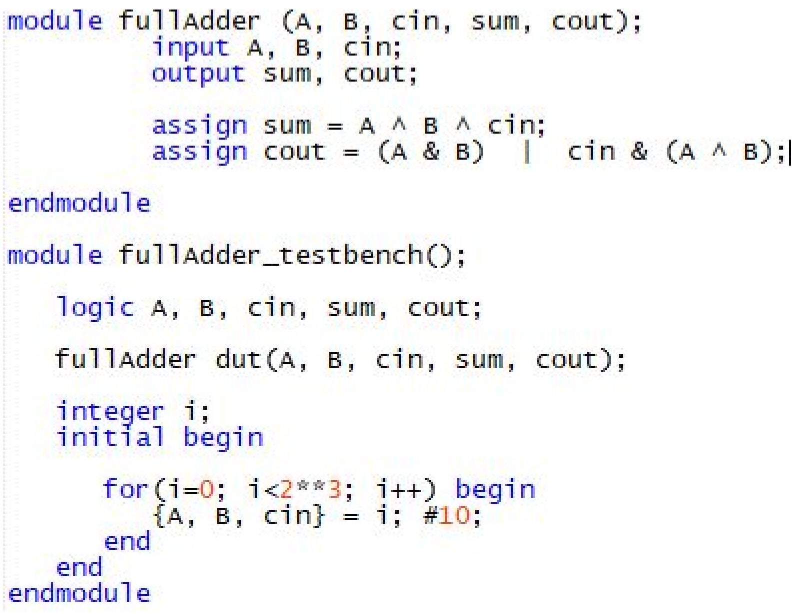

/fullAdder4_testben... 1001
/fullAdder4_testben... 0100
/fullAdder4_testben... 1
/fullAdder4_testben... 1110
/fullAdder4_testben... 0
0110
0111
1000
1001

Figure 4: Segment of 4-bit Adder ModelSim Screenshot

Analysis and Conclusion:

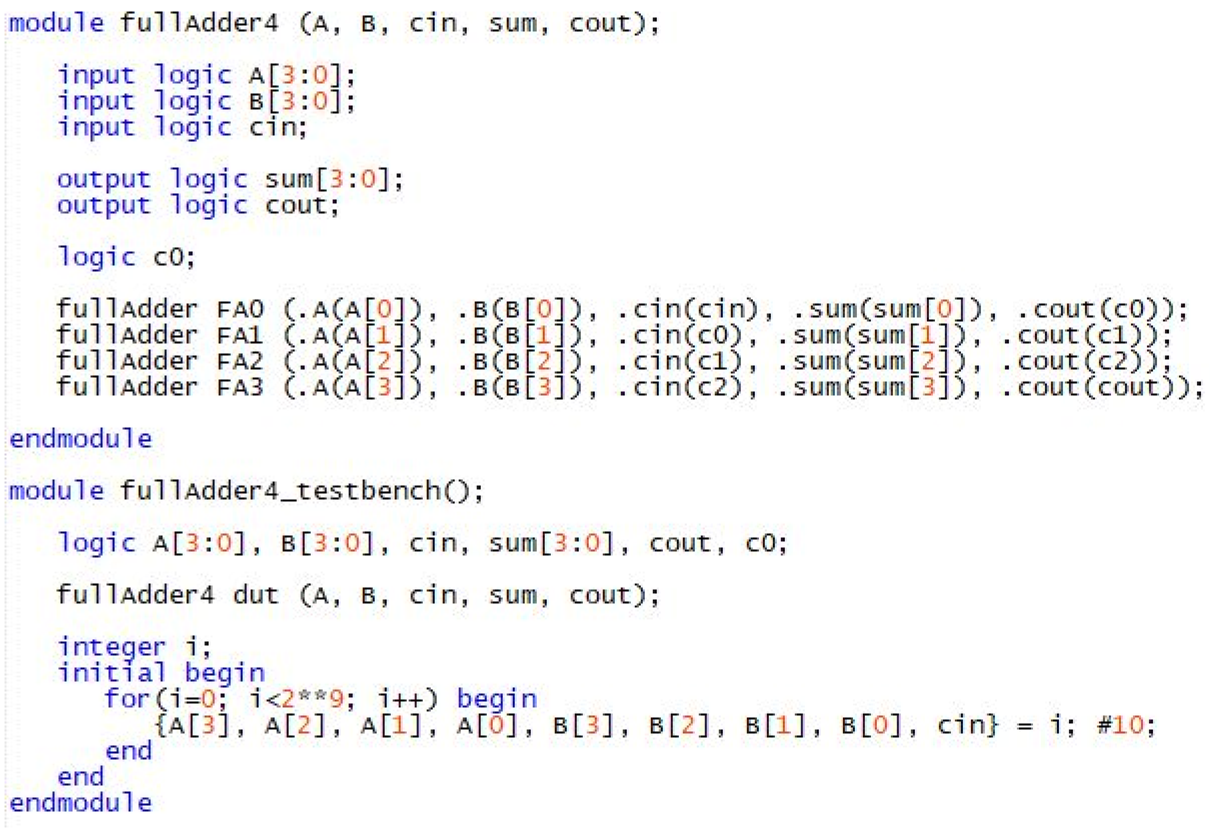
Overall, this lab provided an important introduction to the tools we will be using in EE 271. It will be important to understand the fundamentals of Quartus and Verilog moving forward. The introduction of a full adder is also important for the understanding of logic circuits and how they can be used for practical applications.

Appendix:



```
module fullAdder (A, B, cin, sum, cout);
          input A, B, cin;
          output sum, cout;

          assign sum = A ^ B ^ cin;
          assign cout = (A & B)  |  cin & (A ^ B);

endmodule

module fullAdder_testbench();

   logic A, B, cin, sum, cout;

   fullAdder dut(A, B, cin, sum, cout);

   integer i;
   initial begin

      for(i=0; i<2**3; i++) begin
         {A, B, cin} = i; #10;
      end
   end
endmodule
```

Code 1: Full Adder



```
module fullAdder4 (A, B, cin, sum, cout);

   input logic A[3:0];
   input logic B[3:0];
   input logic cin;

   output logic sum[3:0];
   output logic cout;

   logic c0;

   fullAdder FA0 (.A(A[0]), .B(B[0]), .cin(cin), .sum(sum[0]), .cout(c0));
   fullAdder FA1 (.A(A[1]), .B(B[1]), .cin(c0), .sum(sum[1]), .cout(c1));
   fullAdder FA2 (.A(A[2]), .B(B[2]), .cin(c1), .sum(sum[2]), .cout(c2));
   fullAdder FA3 (.A(A[3]), .B(B[3]), .cin(c2), .sum(sum[3]), .cout(cout));

endmodule

module fullAdder4_testbench();

   logic A[3:0], B[3:0], cin, sum[3:0], cout, c0;

   fullAdder4 dut (A, B, cin, sum, cout);

   integer i;
   initial begin
      for(i=0; i<2**9; i++) begin
         {A[3], A[2], A[1], A[0], B[3], B[2], B[1], B[0], cin} = i; #10;
      end
   end
endmodule
```

Code 2: 4-bit Adder