Lab 8: Carry-Ripple Addition III

Adam Sumner

Illinois Institute of Technology

ECE 429-01

Lab Date: November 2nd, 2015 **Due Date:** November 11th, 2015

1 Introduction

The purpose of this lab is to finalize the design of an adder circuit by combining both the previously designed layout and schematic of a one bit adder from previous labs. Specifically, this lab will tackle building a 4-bit adder circuit. This circuit will then be checked for its accurate functionality using LVS and verilog.

2 Theory/Pre-Lab

2.1 Theory

An adder circuit is one that is able to take two binary inputs and accurately calculate the sum. To implement this via circuitry, addition is done bit by bit. As discussed in the previous two experiments a mirror adder circuit design is an efficient way to perform one bit addition with carry. By combining 4 one bit adders, it's possible to add 4 bit numbers together. This is shown in Figure 1.

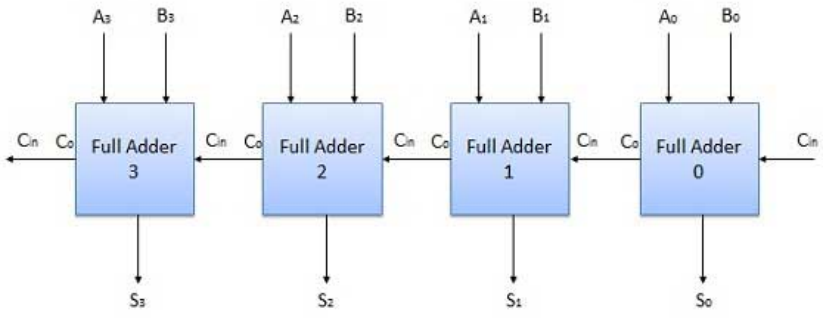

Figure 1: Four Bit Adder Diagram

2.2 Pre-Lab

The pre-lab involved reviewing the carry-ripple architecture discussed in Lab 6. The following two sets of inputs were also translated:

- 11+5
- 4+10

Their translation is shown in Figure 2.

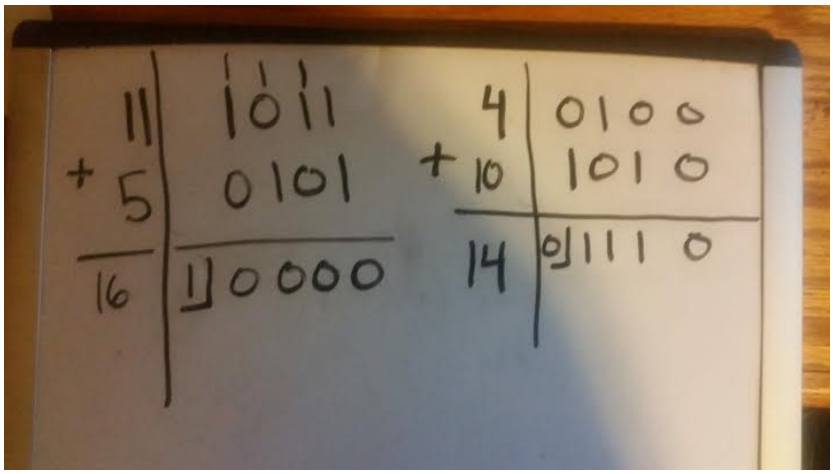

Figure 2: Pre-Lab Translation

3 Implementation

3.1 Schematics

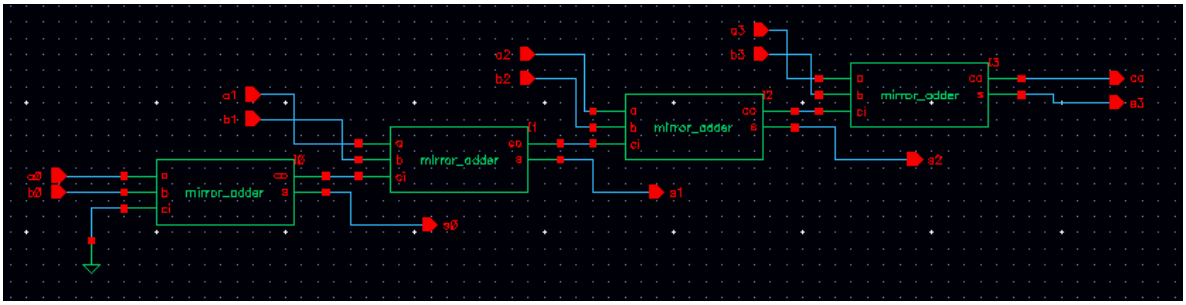

Figure 3: 4-bit Adder Schematic

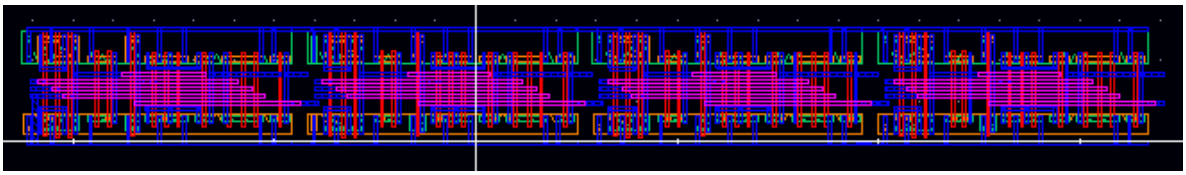

Figure 4: 4-bit Adder Layout

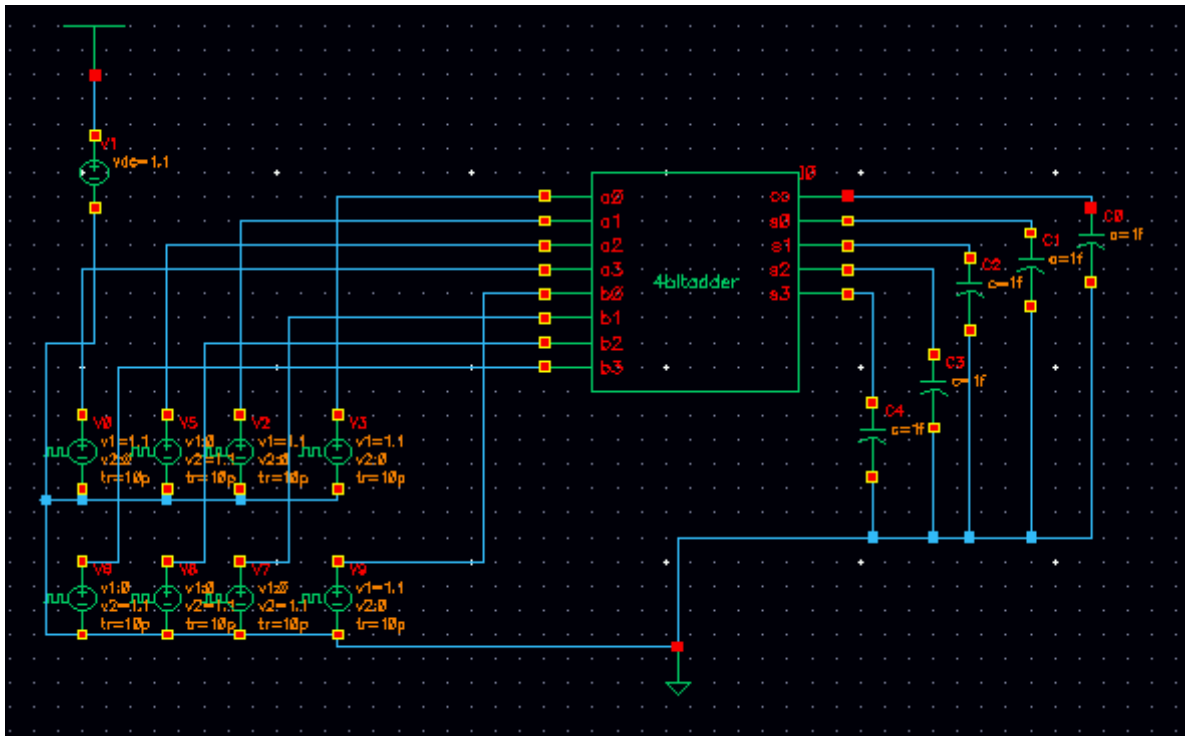

Figure 5: 4-bit Adder Test Circuit ($11 + 5 \rightarrow 4 + 10$)

```
1  module adder_4bit(A,B,S,CO);
2    input [3:0] A,B;
3    output [3:0] S;
4    output CO;
5    assign {CO,S}={1'b0,A}+{1'b0,B};
6  endmodule
```

Figure 6: Verilog Code for 4-bit Adder Verification

```
1   module stimulus;
2     reg [3:0] A,B;
3     wire [3:0] S;
4     wire CO;
5
6     adder_4bit a(A,B,S,CO);
7
8     initial
9     begin
10      #10 A=4'b0000; B=4'b0000;
11      #5 $display("%d+%d=%d %b", A,B,S,CO);
12
13      #10 A=4'b1011; B=4'b0101;
14      #5 $display("%d+%d=%d %b", A,B,S,CO);
15
16      #10 A=4'b0100; B=4'b1010;
17      #5 $display("%d+%d=%d %b", A,B,S,CO);
18
19      #10 A=4'b0001; B=4'b0001;
20      #5 $display("%d+%d=%d %b", A,B,S,CO);
21
22      #10 A=4'b0001; B=4'b0111;
23      #5 $display("%d+%d=%d %b", A,B,S,CO);
24
25      #10 A=4'b1000; B=4'b0111;
26      #5 $display("%d+%d=%d %b", A,B,S,CO);
27    end
28  endmodule
29
```

Figure 7: Verilog Code for 4-bit Adder Verification

3.2 Procedure

The procedure of this lab involved first constructing the schematic and layout of the 4-bit adder. This is shown in Figures 3 and 4. LVS was then run to verify that they are both equivalent. After this, two verilog files shown in Figures 6 and 7 were created and verified using the command:

```
verilog adder4\ test.v adder4.v
```

Once this was done, ESP was run to check that the schematic functionality was the same of the verilog functionality. Once done, a test circuit was created to simulate 11+4 and 4+10. This is shown in Figure 5. Once all the functionality of the adder was

verified, the delay was calculated between the CO signal and S signals with the input A signals.

3.3 Results

```
Compiling source file "adder4_test.v"
Compiling source file "adder4.v"
Highest level modules:
stimulus

 0+ 0= 0 0
11+ 5= 0 1
 4+10=14 0
 1+ 1= 2 0
 1+ 7= 8 0
 8+ 7=15 0
0 simulation events (use +profile or +listcounts option to count)
CPU time: 0.0 secs to compile + 0.0 secs to link + 0.0 secs in simulation
```

Figure 8: Verilog Verification

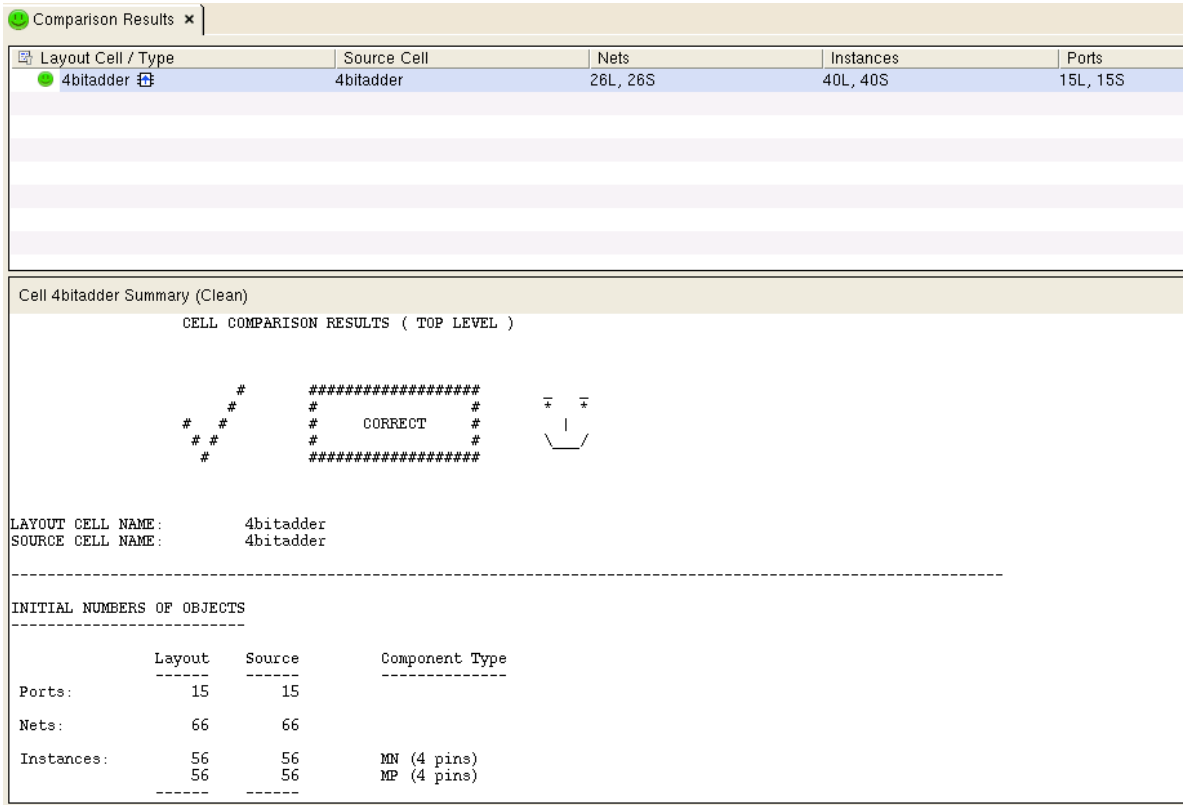

Figure 9: LVS Verification

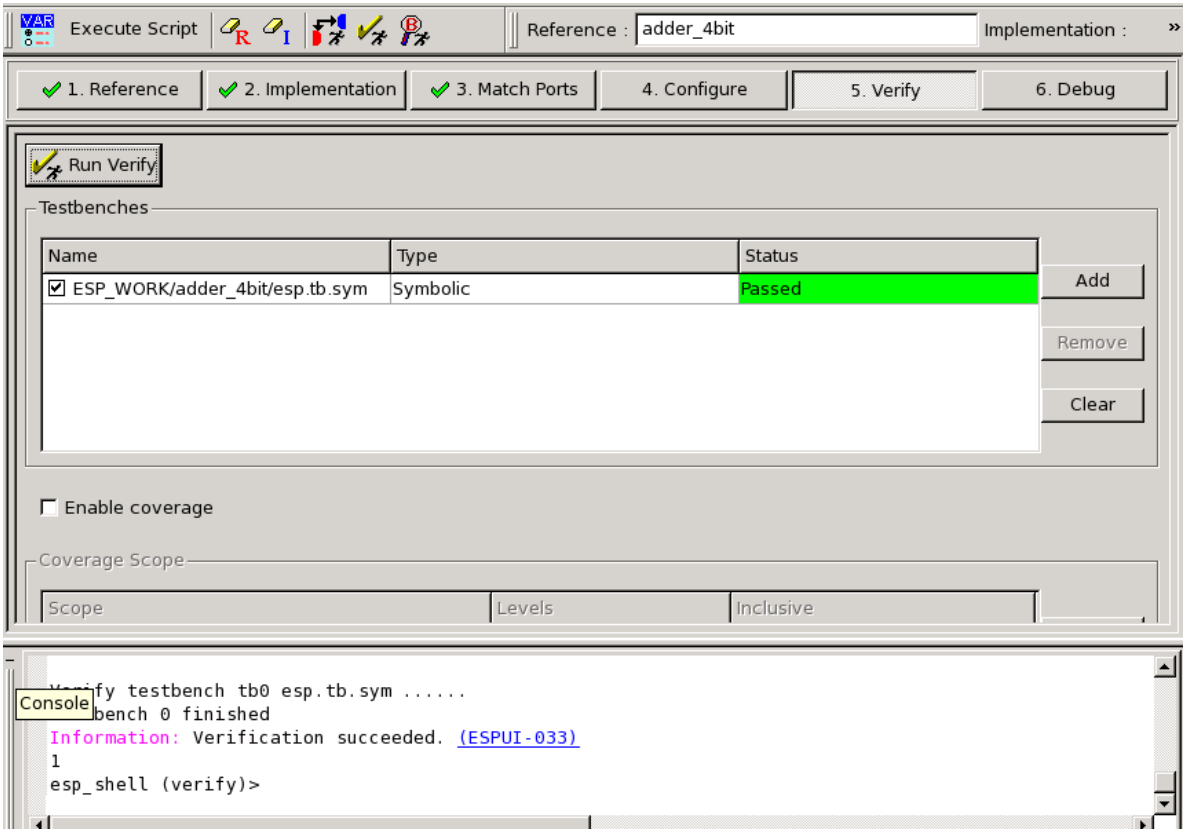

Figure 10: Equivalence Checking using ESP

3.4 Discussion

4 Conclusions