Lab 3 Submission

**5-Bit Ripple Carry Adder (RCA)**

CPE 133 - 03

Michael Hegglin

Jonathan Skelly

**Executive Summary (long, will not do again, Professor said its fine)**

The objectives of this lab were to design a 5-bit ripple carry adder, strengthen our knowledge of the software and hardware being used this quarter, and to show the addition of N-bit inputs (in this case N is 5) to add inputs using full adders together. Using Verilog, we created a circuit that would take 10 one-bit inputs, add them, and output 5 one-bit sums and 1 one-bit carry out using the ripple adder. The circuit was uploaded to a Digilent board, where each input was tied to a switch, and each output would light an LED when its value was “1”. To test that the circuit worked correctly, the assigned switches were flipped. When one switch was flipped, a single LED above on the right side of the board would turn on, signifying equal “sum” outputs of “1” for the first full adder in the ripple adder. When two switches were flipped on opposite sides, the second rightmost LED would turn on while the rightmost LED would go dark, signifying the sum of the second full adder was 1 and the sum of the first full adder being 0 in the ripple adder. This could be continued to show multiple sum outputs and a carry-out. These results were correct, as verified by doing the Boolean equations by hand. This lab produced a fully functional ripple carry adder, and taught through design how a ripple carry adder adds inputs together.

BBD of 5-bit RCA

**Questions**:

**1. Briefly describe the two main attributes of modern digital design**

* The two main attributes of digital design are that everything is hierarchical and modular.

**2. Briefly describe why is it a good idea to avoid modifying previously designed modules in your new design.**

* It is a good idea to not change designs that have already been created because they have already been checked over to make sure they are in working order. If you already checked them and they already work, changing them with only cause errors.

**3. In your own words, briefly but completely explain why the circuit in this lab activity is referred to as a ripple carry adder.**

* This is referred to as a ripple carry adder due to the fact that one of the inputs to the next adder, is the output of the carry-over from the last full adder. In this sense the carry-outs “ripple” down the line of adders.

**4. If you needed to extend the RCA from this lab activity to a 10-bit RCA by using a structural model with two 5-bit RCAs, what changes would you need to apply to the 5-bit RCA?**

* You would simply combine two 5-bit RCAs by putting the carry-out of the last full adder in the sequence to the first full adder of the next RCA. This would then link the RCAs to be used. In the code this would be extending the inputs (a, b) to [9:0] and the output (sum) to [9:0]. You would then connect the LEDs to the sum for the output. This would then increase the amount of input and outputs for 10-bits.

**5. How many rows were there be in a truth table for a 32-bit RCA? Would it be feasible to design a 32-bit RCA using a truth table?**

* It would not make sense to use a truth table for this. There would be 2^64 number of rows.

**6. How many logic gates would it require to implement the 5-bit RCA using discrete logic? For this problem, assume the logic is in reduced form, meaning not in standard SOP.**

* A half adder requires 2 gates. A full adder requires 6 gates. A 5-bit ripple carry adder requires 4 full adders and 1 half adder. Therefore, a 5-bit ripple carry adder takes 26 gates.

**7. Write a formula in closed form that describes the number of gates in a RCA as a function of the bit- width of the RCA. Recall that the LSB of the RCA is a HA. For this question, assume the FA and HA are in reduced form (not in standard SOP form).**

* # gates = 6(n - 1) + 2\*(1)

**8. For a RCA, the result could be available immediately, or the result could be delayed. Describe a case where the result is available immediately and also describe a case where the delay is the “worst case”. State how long the worst case is in terms of “gate delays”.**

* If the result is available immediately then there would be no carry-over to the next full adder in the ripple carry adder. The sum of the first adder would be your answer. An case with no delay would be feeding 00000 into a 5-bit RCA. The worst case is the inputs running through every gate in the ripple carry adder. This is due to the fact that one full adder must process before the next goes. A case with worst delay would be running 11111. This time delay for a 5-bit ripple carry adder would be 11 gate delays.

**9. Do the various module instantiations in a HDL model operate in a concurrent manner? Briefly explain why or why not.**

* They do occur concurrently. When modules are instantiated, Vivado takes blocks concurrently and puts them together. Therefore, the instantiations are concurrent.

**10. Briefly describe the notion of concurrency in digital circuit design.**

* It means that multiple statements in say verilog can be executed at the same time to perform operations.

**11. Consider two different HDL models that are functionally equivalent, in particular, they are mostly the same except for the notion that one used structural modeling and the other implemented a similar set of modules but did not use structural modeling. Would expect the synthesized circuit based on these models to use the equivalent amount of resources or will one approach use more or less resources. Support your answer with intelligent commentary.**

* They should end up taking the same amount of resources due to the fact that there are still the same amount of gates to pass through. Structure modeling is better because modules are reusable.

**Design Problem:**



BBD of 10-bit, 4-input RCA

Next Lower-Level diagram of 10-bit 4-input RCA circuit with Sum and Valid outputs.

**Source Code:**

timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 10/03/2018 11:47:10 AM

// Design Name:

// Module Name: lab3

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module half\_adder(a, b, sum, co);

input a, b;

output sum, co;

assign sum = (~a & b) | (a & ~b);

assign co = (a & b);

endmodule

module full\_adder(a, b, c, sum\_sop, sum\_pos, co\_sop, co\_pos);

input a, b, c;

output sum\_sop, sum\_pos, co\_pos, co\_sop;

assign sum\_sop = (~a & ~b & c) | (~a & b & ~c) | (a & ~b & ~c) | (a & b & c);

assign sum\_pos = (a | b | c) & (a | ~b | ~c) & (~a | b | ~c) | (~a | ~b | c);

assign co\_sop = (~a & b & c) | (a & ~b & c) | (a & b & ~c) | (a & b & c);

assign co\_pos = (a | b | c) & (a | b | ~c) & (a | ~b | c) & (~a | b | c);

endmodule

module lab3(a, b, sum, carry\_out);

input [4:0] a, b;

output [4:0] sum;

output carry\_out;

wire [3:0] carry\_o;

half\_adder half1(

.a (a[0]),

.b (b[0]),

.co (carry\_o[0]),

.sum (sum[0])

);

full\_adder full\_1(

.a (a[1]),

.b (b[1]),

.c (carry\_o[0]),

.co\_sop (carry\_o[1]),

.sum\_sop (sum[1])

);

full\_adder full\_2(

.a (a[2]),

.b (b[2]),

.c (carry\_o[1]),

.co\_sop (carry\_o[2]),

.sum\_sop (sum[2])

);

full\_adder full\_3(

.a (a[3]),

.b (b[3]),

.c (carry\_o[2]),

.co\_sop (carry\_o[3]),

.sum\_sop (sum[3])

);

full\_adder full\_4(

.a (a[4]),

.b (b[4]),

.c (carry\_o[3]),

.co\_sop (carry\_out),

.sum\_sop (sum[4])

);

endmodule