Lab 0: Full Adder on FPGA

Samantha Young, Paige Pfenninger, Lauren Pudvan

9/25/18

1 Simulation Waveform

Below is the schematic we coded into Verilog for a 1 bit full adder.

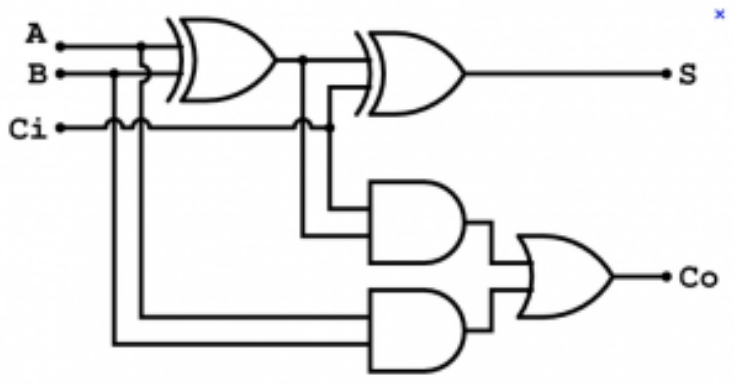

Figure 1: Schematic of a 1 bit full adder from www.circuitstoday.com

We connected 4 of these units in order to create a 4 bit adder.
Below is the waveform of the 4 bit adder.

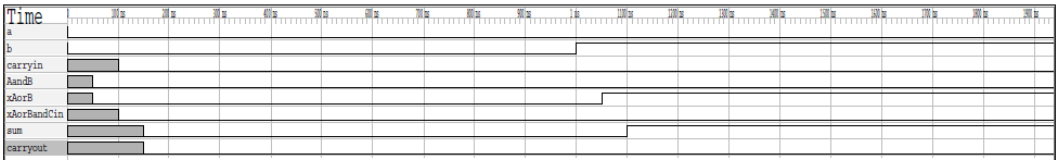

Figure 2: The waveform propagation of a full adder when B changes from 0 to 1

The waveform in figure 2 shows the delay propagation after B is changed from 0 to 1. When B is changed from 0 to 1, first the output of the Xor gate between A

and B goes high and then that signal causes the sum to go high which is the output of an Xor gate of carry in and the Xor of A and B.

Carryout and the outputs of the other gates do not change because we only changed the value of B. If we were to show the propagation of a signal where A and carry in were changed, the outputs of other gates would change.

The following table shows the worst case scenario for propagation delay for our circuit:

	Cout	Sum
A	3	2
B	3	2
Cin	2	1

This table assumes that every gate has a constant delay. The table basically shows the maximum number of gates a signal has to go through to get to the output.

2 Test Case Strategy

Our strategy for testing was to test various cases that were more likely to fail. We tested the 2 biggest numbers together, the two lowest numbers, a couple cases where the operator was zero, a couple where the result was a zero, some positive and negative number combinations that would and would not have carryout, a couple large positive numbers to get overflow, a couple very low numbers to get overflow, and various cases that would not have carryover.

Below is the designed test case suite. It includes 16 test cases that involve different scenarios in order to get different combinations of overflow and carryout.

A	B	Sum	Carry Out	Overflow
0001	0000	0001	0	0
0000	1100	1100	0	0
0111	0111	1110	0	1
1000	1000	0000	1	1
1111	0111	0110	1	0
0011	1101	0000	1	0
1111	1111	1110	1	0
0001	0001	0010	0	0
0010	0001	0011	0	0
0100	0101	1001	0	1
1110	1101	1011	1	0
1001	1000	0001	1	1
1011	1010	0101	1	1
1101	0111	0100	1	0
0001	1100	1101	0	0
0010	1010	1100	0	0

Our test cases never failed because we were correct the first time!

3 FPGA Testing

In order to test on the FPGA we set each of the two 4-bit numbers that we wanted to add using the 4 switches, where each switch corresponded to a bit. For the first number we set the switches then clicked the first button to save the number. Then we flipped the switches to represent the second number and clicked the second number to set it. When clicking the third button, the LEDs lit with the sum of the two numbers. When we clicked the fourth button, the first and second LEDs corresponded to carryout and overflow.

Below is a video of two examples being set on the FPGA. We tried out the examples from the truth table above to ensure that the switches, buttons and lights worked as expected. In these examples the the sums, carryout, and overflow are all correctly computed. The video includes examples with overflow and carryout as well as and example without. Please see this video: `https://www.youtube.com/watch?v=YhgLhUoi1SY`

4 Vivado Summary

Below is a summary of the statistics from the synthesized design from Vivado.

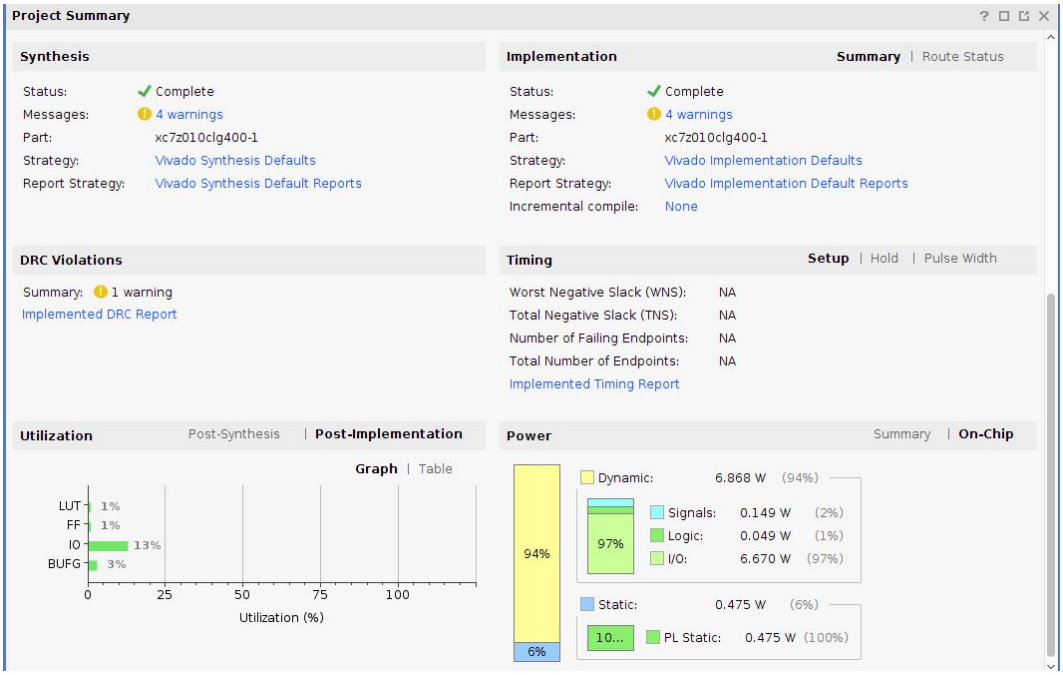

Figure 3: Summary in Vivado