Lab 0: Full Adder on FPGA

Andrew Pan, Robert Siegel, Kim Winter

September 2017

1 4-bit Adder Design

Using the full adder design that we designed in our homework, we were able to design an adder that incorporated the full adder design into a design that added two 4-bit inputs, using two's complement.

Our initial design involved an approach that integrates the principles of basic addition, where each column, starting with the right-most column of the bit is added. The overflow, or "carryout", then becomes an input, or "carryin", to another full adder, which adds the second right-most column and so forth. This was able to give us an output, given no overflows.

Next, we had to adjust our design and flag results when they overflowed, meaning that our answer would not be correct. We knew that overflow could only occur if the two addends were opposite signs. Since the sign is dictated by the left-most bit, we compared these (e.g. $(a_4 b_4)+(\bar{a}\bar{b})$).

Then, we had to test if an overflow actually occurred, meaning that two positives added to a negative, or two negatives added to a positive. So, if the first bit of the sum is not equal to the first digit of a or b, then this means that a sign has switched. This indicates that two negative integers summed to a positive integer, or two positive integers summed to a negative integer. This will send a flag, meaning that our circuit detected the overflow. Our final overflow logic diagram is shown in Figure 1.

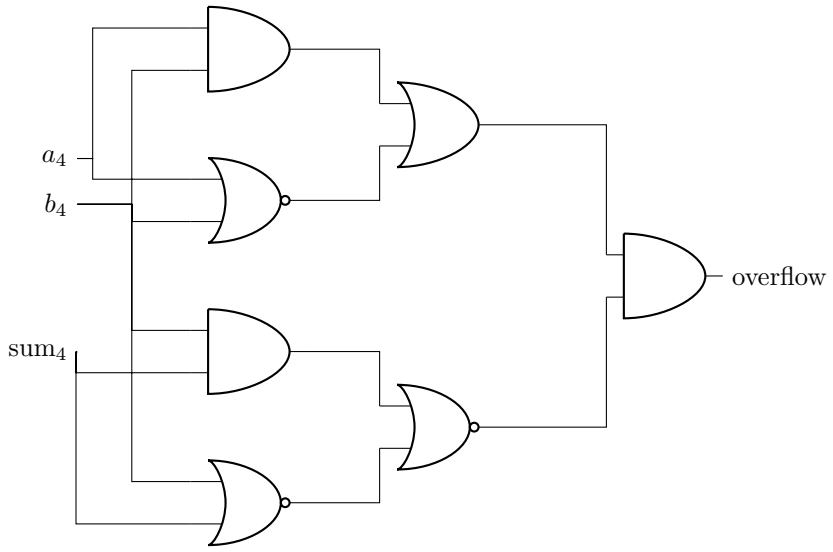


Figure 1: Our final overflow logic gate diagram

2 Test Bench

We chose twelve categories of test cases to run that would cover the possible edge cases involving overflows and carryouts that our adder could encounter. We then further developed these into sixteen rigorus test cases for our system. These sixteen cases fell into the following categories:

- Two positive addends with overflow
- Two positive addends without overflow
- Two negative addends without overflow
- Two negative addends with overflow
- One positive addend p and one negative addend n for which $p = |n|$
- One positive addend p and one negative addend n for which $p > |n|$ with carryout
- One positive addend p and one negative addend n for which $p < |n$ with carryout
- One positive addend p and one negative addend n for which $p > |n|$ without carryout
- One positive addend p and one negative addend n for which $p < |n|$ without carryout

- One zero addend and one positive addend
- One zero addend and one negative addend
- Two zero addends

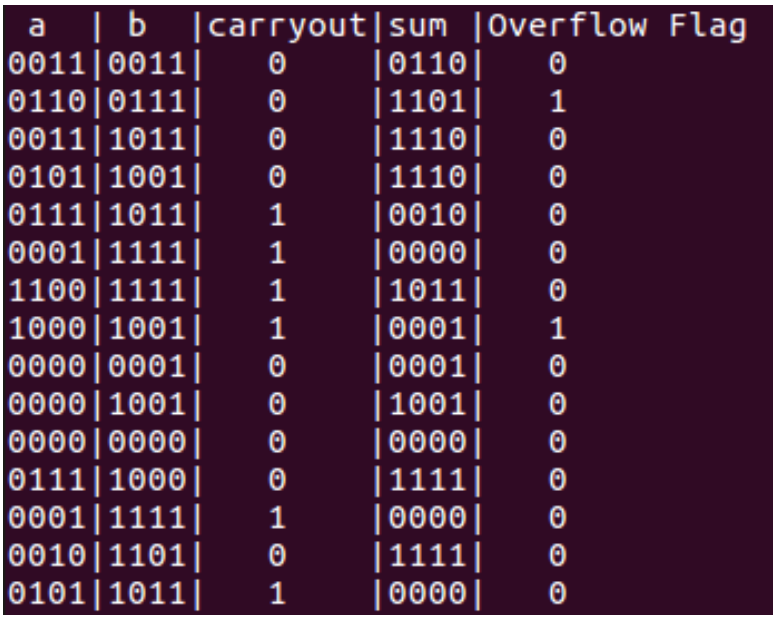
To develop these test cases, we considered what possibilities existed that combined adding positive integers, negative integers, and zero. In addition, we considered which of these combinations could result both with and without overflow, as well as with and without carryout. For example, it is not possible to add two negative integers without a resultant carryout under this system, so we assumed that in our test cases. However, two negative integers could lead to a result with or without overflow, so it was necessary to check both cases.

We vigorously checked the logic of our design throughout the process, and we were happy to find that this resulted in all of our tests passing on the first attempt.

3 FPGA

3.1 FPGA Test Cases

We ran the sixteen test cases used in our test bench on our FPGA. Ten of these were based on the original basic and edge cases we tested in ModelSim. The code used to test 4-bit full adder was the provided wrapper file, and the LED's of the FPGA were checked to ensure they were the expected output for each of our test bench cases. Our final sixteen test cases are shown in Figure 2.



a	b	carryout	sum	Overflow Flag
0011	0011	0	0110	0
0110	0111	0	1101	1
0011	1011	0	1110	0
0101	1001	0	1110	0
0111	1011	1	0010	0
0001	1111	1	0000	0
1100	1111	1	1011	0
1000	1001	1	0001	1
0000	0001	0	0001	0
0000	1001	0	1001	0
0000	0000	0	0000	0
0111	1000	0	1111	0
0001	1111	1	0000	0
0010	1101	0	1111	0
0101	1011	1	0000	0

Figure 2: Our final 16 test cases for our FPGA

None of our test were able to cause failures on the FPGA, and we recorded a video of one test on our FPGA: Video of test case on FPGA

4 Results

4.1 Synthesized Design Summary Statistics

Route Status Report:
Design Route Status
logical nets: 42
nets not needing routing: 15
internally routed nets: 15
routable nets: 27
fully routed nets: 27
nets with routing errors: 0

Resource	Utilization	Available	Utilization %
LUT	6	17600	0.03
FF	9	35200	0.03
IO	13	100	13.00

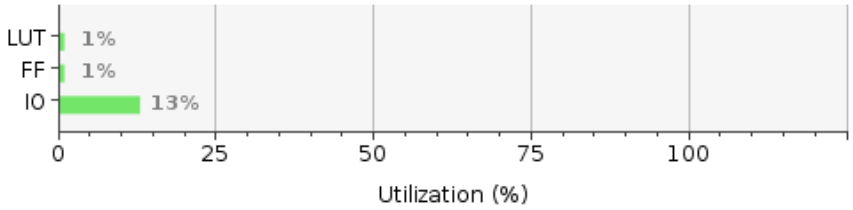


Figure 3: Resource Utilization in our FPGA

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.113 W**
Junction Temperature:	**26.3 °C**
Thermal Margin:	58.7 °C (5.0 W)
Effective θJA:	11.5 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

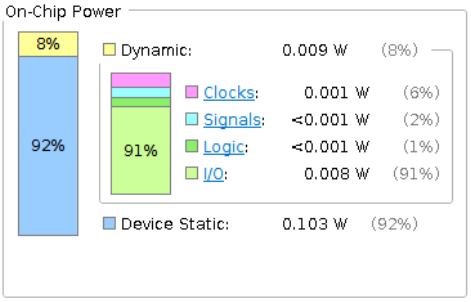


Figure 4: A power analysis of our FPGA

4.2 Waveforms

Below is a graph of our waveforms for the full adder, holding inputs until all of the adder outputs have stabilized. From the graph, it appears that the longest propagation delay occurs for the sum outputs, and is 300 base time units (nanoseconds in this case). This is expected, especially given the fact that our carryout takes less time to propagate than our sum, which is consistent with the circuit that we designed.

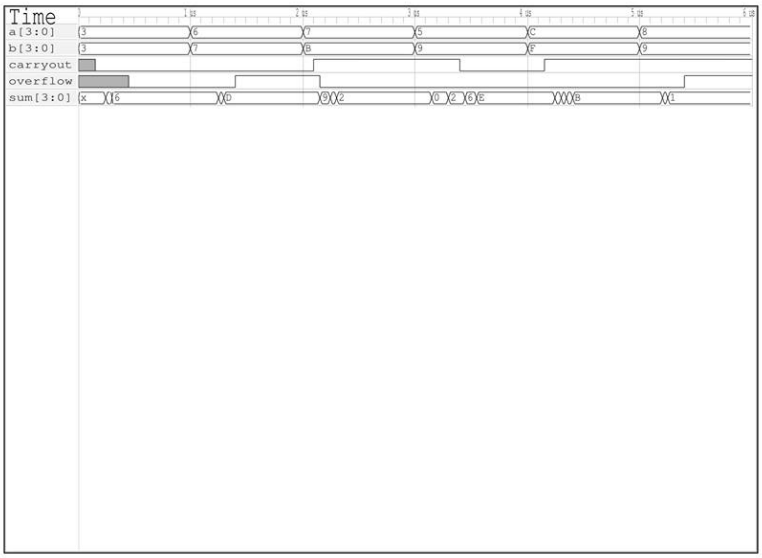


Figure 5: How our circuit behaves in time, using GTK Wave