CompArch: Lab 0

Zoe Fiddler, Ezra Varady

October 2, 2015

1 Adder Design

For the 4bit full adder we reused the design of a 1 bit full adder from a previous assignment. We then hooked up 4 in series, connecting the carryout of the previous one into the carryin of the following one. The carryout of the final 1 bit adder is the carryout of the whole system. Overflow is calculated as the carryin to the final 1 bit adder `XOR` the carryout of that same adder.

2 Test Cases

```
# These show addition is working properly in cases with no overflow or carryout
#   A    B   |  S   Cout Ovf |  S   Cout Ovf  Expected
# 0000  0000 | 0000   0   0  | 0000   0   0
# 0001  0000 | 0001   0   0  | 0001   0   0
# 0010  0001 | 0011   0   0  | 0011   0   0
# 0011  0011 | 0110   0   0  | 0100   0   0
# 1100  0011 | 1111   0   0  | 1111   0   0
# 0111  1000 | 1111   0   0  | 1111   0   0
# ---------------------------------------------------------------
# These are cases with carryout but no overflow
#   A    B   |  S   Cout Ovf |  S   Cout Ovf  Expected
# 1110  1101 | 1011   1   0  | 1011   1   0
# 1110  1110 | 1100   1   0  | 1101   1   0
# ---------------------------------------------------------------
# Inverses sum to 0
#   A    B   |  S   Cout Ovf |  S   Cout Ovf  Expected
# 0010  1110 | 0000   1   0  | 0000   1   0
# 1010  0110 | 0000   1   0  | 0000   1   0
# ---------------------------------------------------------------
# These are cases with overflow but no carryout
#   A    B   |  S   Cout Ovf |  S   Cout Ovf  Expected
# 0101  0100 | 1001   0   1  | 1001   0   1
# 0011  0111 | 1010   0   1  | 1010   0   1
# ---------------------------------------------------------------
# These are cases with overflow and carryout
#   A    B   |  S   Cout Ovf |  S   Cout Ovf  Expected
# 1001  1000 | 0001   1   1  | 0001   1   1
# 1100  1010 | 0110   1   1  | 0110   1   1
```

Figure 1: Test bench for 4 bit full adder

We aimed to create test cases for each of the different black box behaviors as well as propagation through the adder. This includes testing proper sum is found in cases with no carryout or overflow, carryout and no overflow, overflow and no carryout and both carryout

and overflow. The test bench produced in Modelsim can be seen in Figure 1 on the preceding page. No failures were encountered as the first iteration of the adder worked. Figure 2 shows the waveforms during testing. The sum, carryout and overflow can be seen to change and settle some interval after the inputs A and B change.

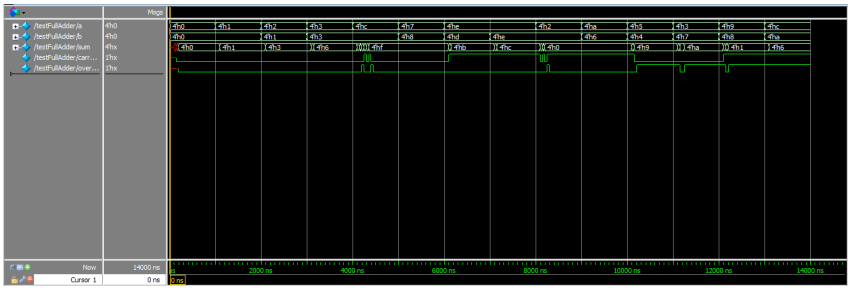

Figure 2: Waveforms showing output response to changing input

3 FPGA

We implemented this full adder on an FPGA and tested it with the same cases we used in our Modelsim test bench. The test cases and results can be seen in Table 1

Table 1: Test cases for FPGA

A	B	SUM	CarryOut	OVF
0000	0000	0000	0	0
0001	0000	0001	0	0
0010	0001	0011	0	0
0011	0011	0110	0	0
1100	0011	1111	0	0
0111	1000	1111	0	0
1110	1101	1101	1	0
1110	1110	1100	1	0
0010	1110	0000	1	0
1010	0110	0000	1	0
0101	0100	1001	0	1
0011	0111	1010	0	1
1001	1000	0001	1	1
1100	1010	0110	1	1

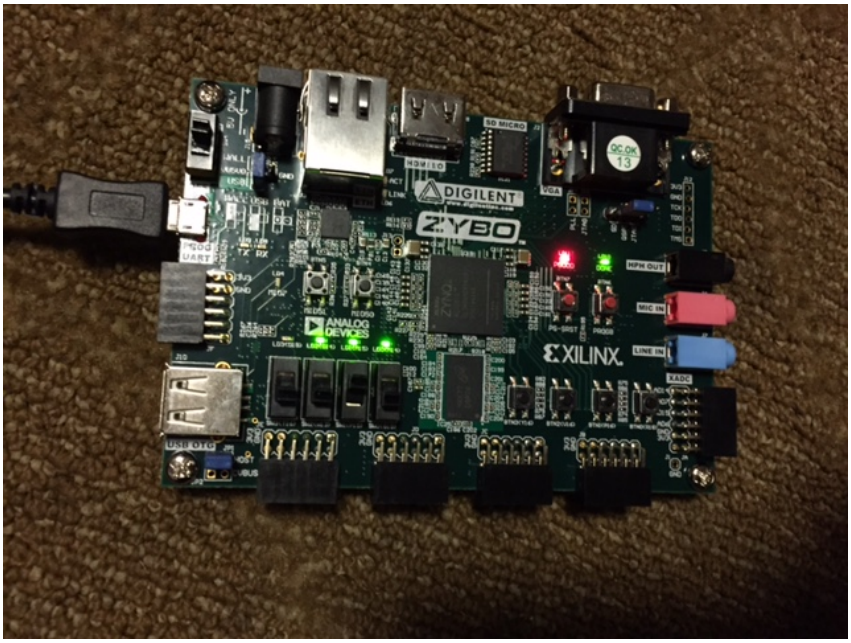

Figure 3: Testing on FPGA

4 Statistics

A 1 bit full adder has a maximum prop delay of 3 gates, which translates to a maximum prop delay of 13 gates for the 4 bit adder, including the `XOR` for overflow. In our test cases, with each gate introducing a $50ns$ delay, the maximum delay we saw was $450ns$.

The implementation on the FPGA used 9 flip flops (0.03% of available) and 10 look up tables (0.06% of available). It had a worst negative slack time of $6.93ns$.