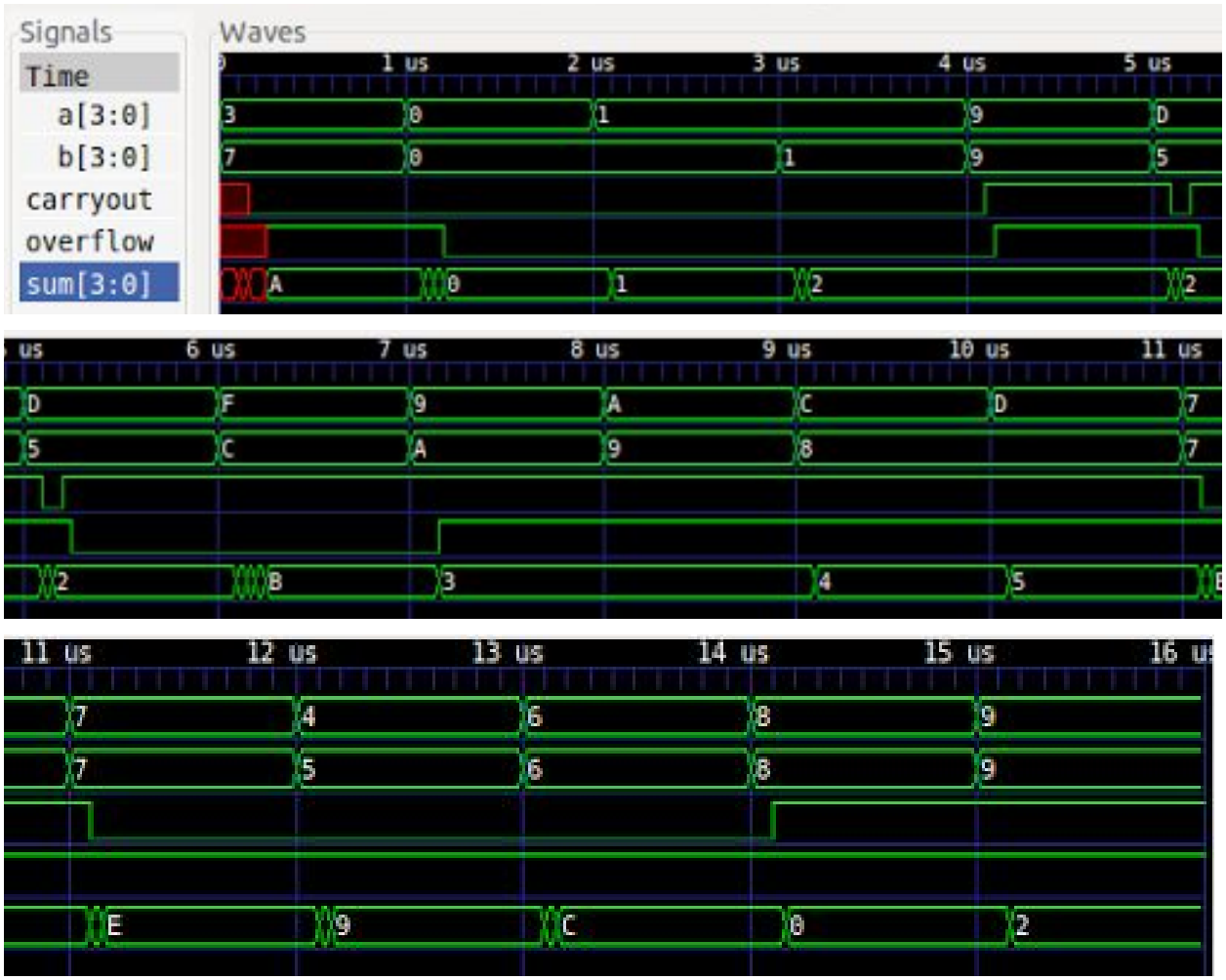
Lab0 Report

Anna Buchele, Apurva Raman

1. Waveform showing the full adder and delay analysis:



Figures 1-3: Waveform of completed adder, sectioned for readability

Above, we can see the waveform of our full 4 bit adder. The values are displayed in hexadecimal. As you can see, the waveform appears to display the correct values for the carryout, overflow, and sum after a short period of adjustment. If the result is the same as it was for the previous set of inputs, as we can see in Figure 2: (9 + A) and (A+9), there is no period of adjustment and the results hold. Certain results require a longer period of adjustment, with the sum switching up to four times before setting on the result. However, each of the sets settle on the final, correct answer well before the time limit for the data to be read expires.

Delay Analysis:

	a	b	carryin
sum	2	2	1
carryout	3	3	2

Table 1: Worst Case Number of Gates (For each run of structuralFullAdder)

As we can see in the table above, the worst case number of gates for any variable during a single run of structuralFullAdder is 3, which is the maximum number of gates for a or b to get to the carryout. Since structuralFullAdder is run a total of four times during FullAdder4bit, the worst case number of gates for the whole program is 12. However, we also have one last gate at the end to check for overflow, and that uses both the carryin and the carryout. So, the total number of gates for the worst case of our FullAdder4bit is 13. Since each gate takes 50 units of time, 13 gates will take 650 units of time. Our current model reads the results after 1000 units of time, so that is plenty of time for everything to stabilize.

2. Test Case Strategy:

We started by choosing 4 examples where we focused on whether the sum was the expected value. For these, we used positive numbers for all of the test cases. Three were without overflow, and one was with overflow.

Since we made a signed 4-bit adder, we needed cases that covered both overflow and carryout situations.

For carryout, we looked at two cases where there was a carryout of one and there was overflow and two cases with a carryout with no overflow.

For overflow, we looked at eight cases of two positive numbers added together, resulting in overflow and five cases of two negative numbers added together, resulting in overflow.

3. Test Case Failures:

We had a recurring problem where several of our inputs would get switched to different numbers in the test bench. An example of this was for inputs (a = 1111, b = 1100), in the test bench they would be displayed and added as (a = 0111, b =1100). It was always the most significant bit changed, and sometimes in both a and b or sometimes just one. After much confusion, we finally asked a NINJA. He recommended placing 4’b in front of each of the inputs

(example: a = 4'b1001), to tell the computer that it was a 4-bit number. This solved the problem, and after that our inputs and outputs displayed normally.

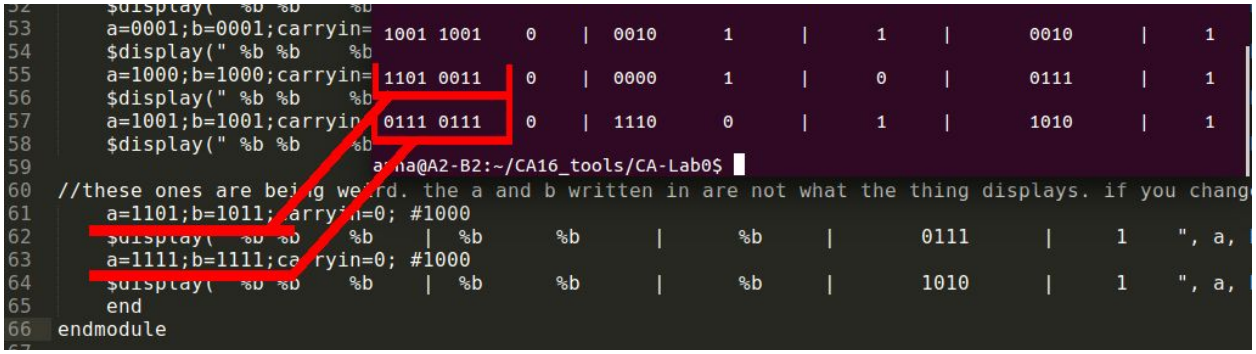


Figure: An example of the test case failure/ changed inputs

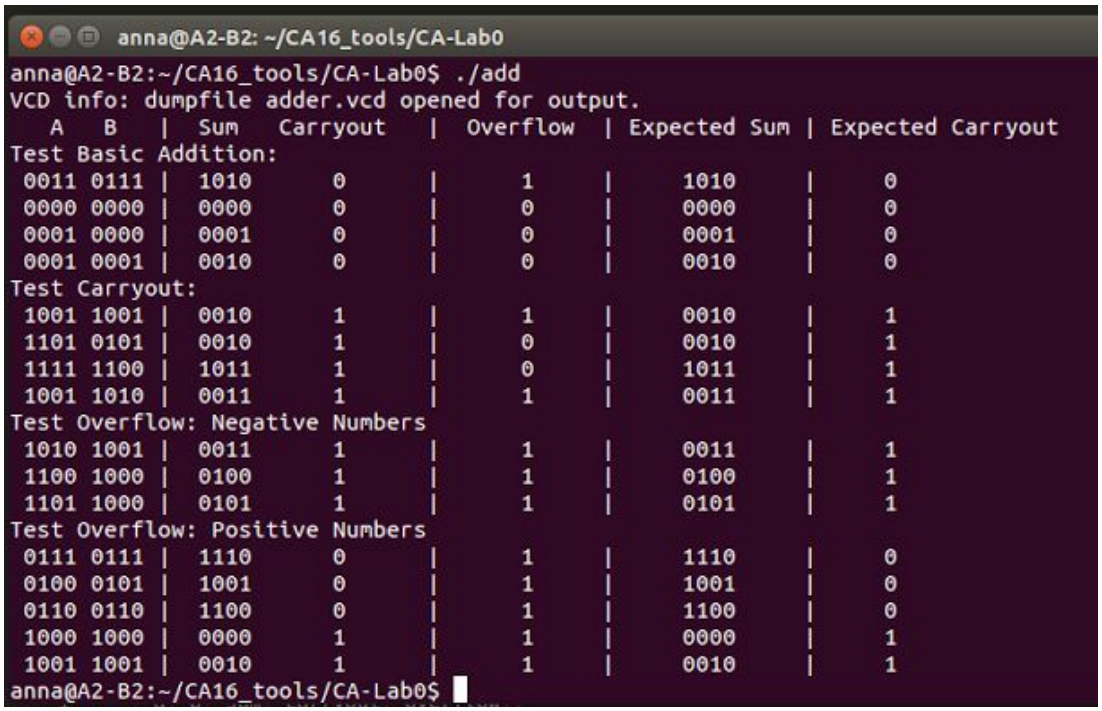


Figure: Our final test bench output

4. Summary of FPGA testing

Once we changed the module names to match the wrapper, performed synthesis, and generated the bitstream, we tried one of each of our test case categories on the FBGA board and made sure the

Videos are on public (/public/Anna-ApurvaCompArch) for the following tests:

Zybo Test: 0111 + 0111 = 1110, overflow = 1, carryout = 0

Zybo Test1: 1000 + 1000 = 0000, overflow 1, carryout 1

Zybo Test2: 1010 + 1001 = 0011, overflow = 1, carryout = 1

5. Summary statistics

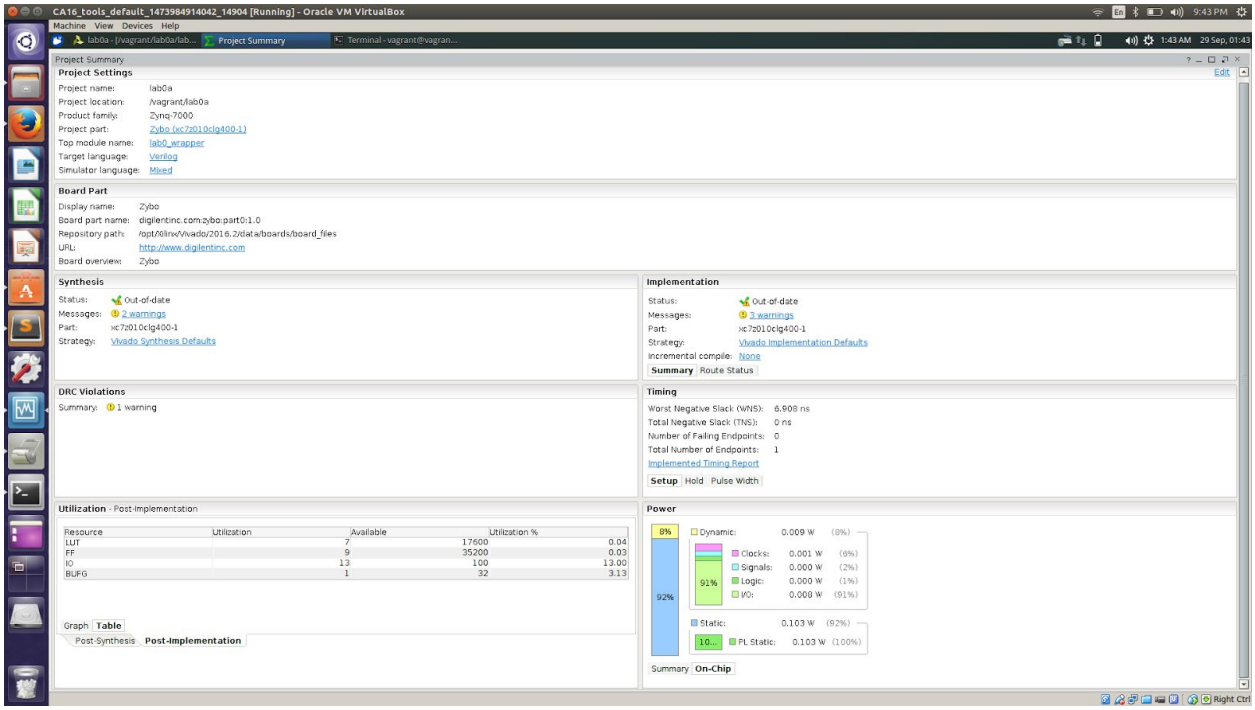
Performance:
The worst negative slack was 6.908 ns, and there was one endpoint. This is because of the one clock indicated in the energy segment, which we think was auto-generated.

Energy:
The total energy consumption was 0.113 W. The clock took 6%, the signals took 2%, the logic took 1%, and the I/O took 91%. This makes sense given that our circuits are very simple and small for this project, so the energy consumption for buttons, switches, and LEDs is much greater.

Area:
The program utilized:

LUTs	7
FFs	9
I/O	13
BUFG	1

This isn't a huge area, but as discussed in class, several lookup table calls can result in larger delays.


CA16_tools_default_1473984914042_14904 [Running] - Oracle VM VirtualBox
Machine View Devices Help
Project Summary
Project Settings
Edit
Project name: lab0a
Project location: /vagrant/lab0a
Product family: Zynq-7000
Project part: Zybo (xc7z010clg400-1)
Top module name: lab0_wrapper
Target language: Verilog
Simulator language: Mixed
Board Part
Display name: Zybo
Board part name: digilentinc.com:zybo:part0:1.0
Repository path: /opt/Xilinx/Vivado/2016.2/data/boards/board_files
URL: http://www.digilentinc.com
Board overview: Zybo
Synthesis
Status: Out-of-date
Messages: 2 warnings
Part: xc7z010clg400-1
Strategy: Vivado Synthesis Defaults
Implementation
Status: Out-of-date
Messages: 3 warnings
Part: xc7z010clg400-1
Strategy: Vivado Implementation Defaults
Incremental compile: None
Summary Route Status
DRC Violations
Summary: 1 warning
Timing
Worst Negative Slack (WNS): 6.908 ns
Total Negative Slack (TNS): 0 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 1
Implemented Timing Report
Setup Hold Pulse Width
Utilization - Post-Implementation
Resource Utilization Available Utilization %
LUT 7 17600 0.04
FF 9 35200 0.03
IO 13 100 13.00
BUFG 1 32 3.13
Graph Table
Post-Synthesis Post-Implementation
Power
8%
92%
Dynamic: 0.009 W (8%)
Clocks: 0.001 W (6%)
Signals: 0.000 W (2%)
Logic: 0.000 W (1%)
I/O: 0.008 W (91%)
91%
Static: 0.103 W (92%)
10...
PL Static: 0.103 W (100%)
Summary On-Chip