Lab 2 - The Only Time You Have To Do Math

Embedded Systems 1
Lab Report 2
Tim Petersen - trp87
3/5/2020

Purpose

In this experiment, we are going to create a simple 4-bit Arithmetic and Logic Unit (ALU). First, a single bit full adder will be designed using basic Boolean logic. Then, a 4-bit ripple-carry adder will be created structurally using the single bit full adder. Finally, the power of operators will be leveraged in order to behaviourally describe a 16 function ALU.

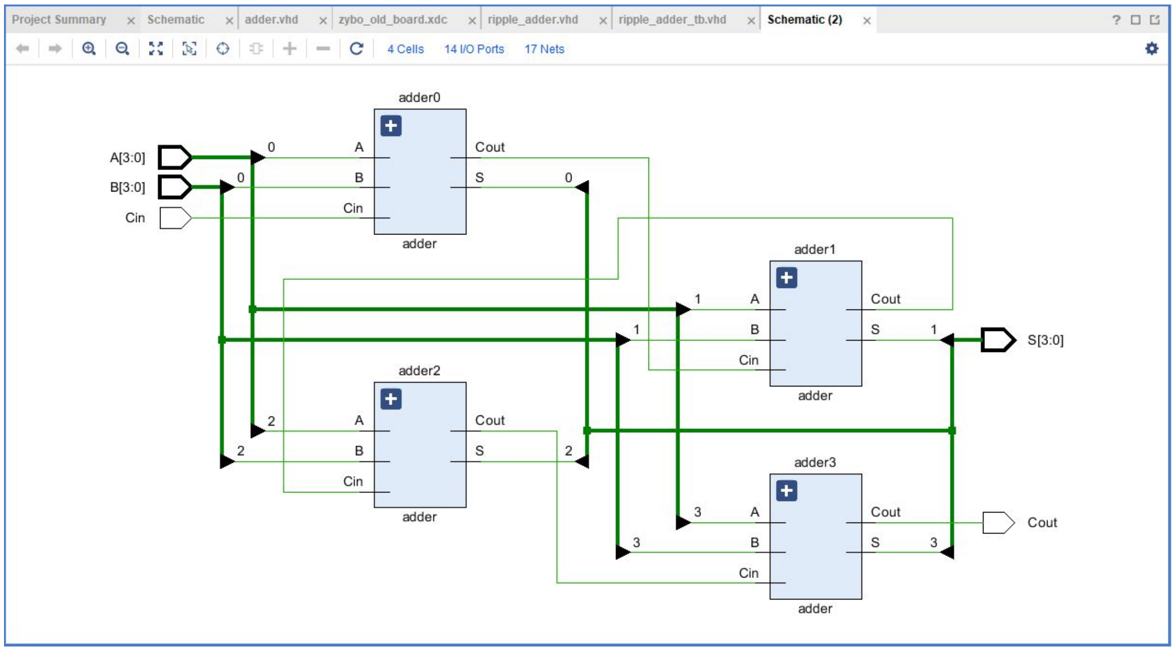
Ripple Adder

Theory of Operation:

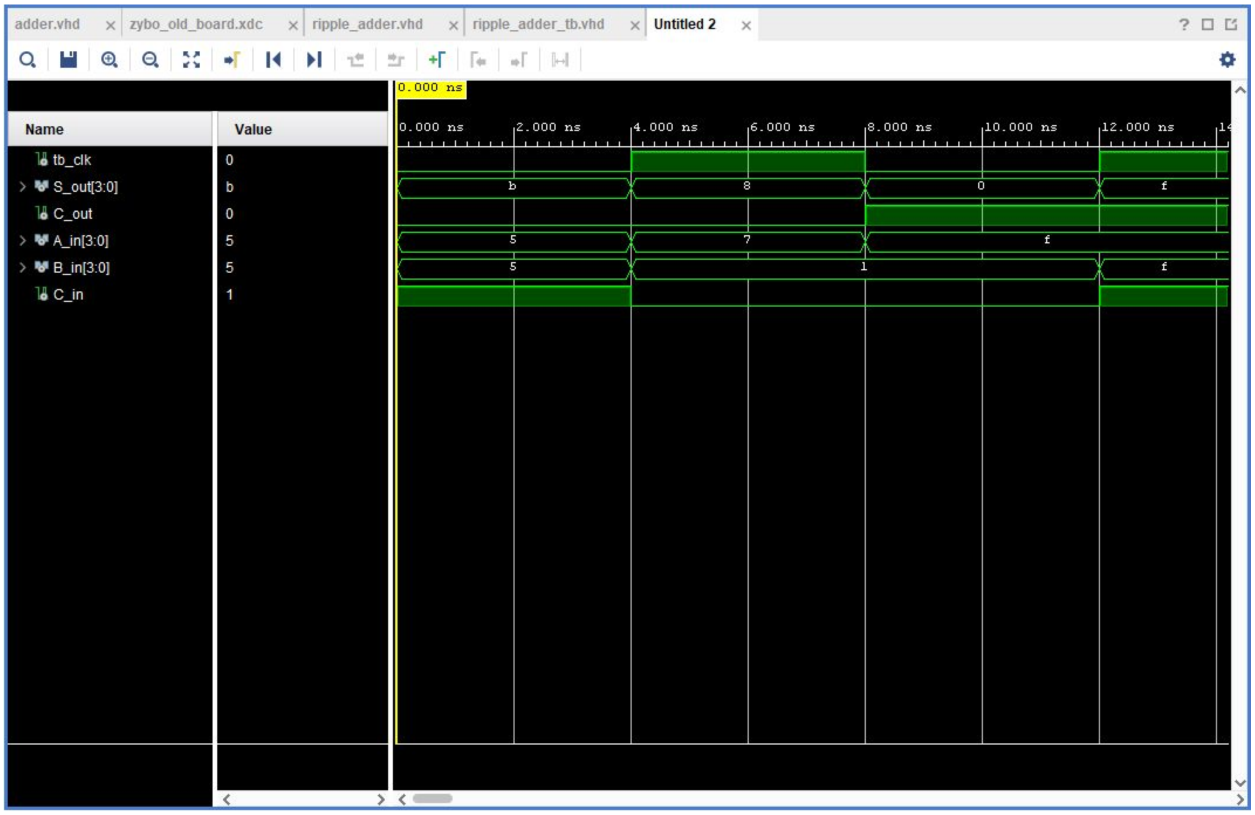
The one bit full adder will add three numbers, A, B, and the carry in bit. If the answer is over the number of bits the output can represent the carry out bit will be one. To implement a 4 bit ripple adder, four single bit adders were put in parallel with each carry out bit going into the next carry in. This should theoretically be able to add 2 numbers that are less than 15. If the sum exceeds 15 the carry out bit will be 1

Code: on Github (trp87-rutgers)
Test Bench Code: on Github (trp87-rutgers)

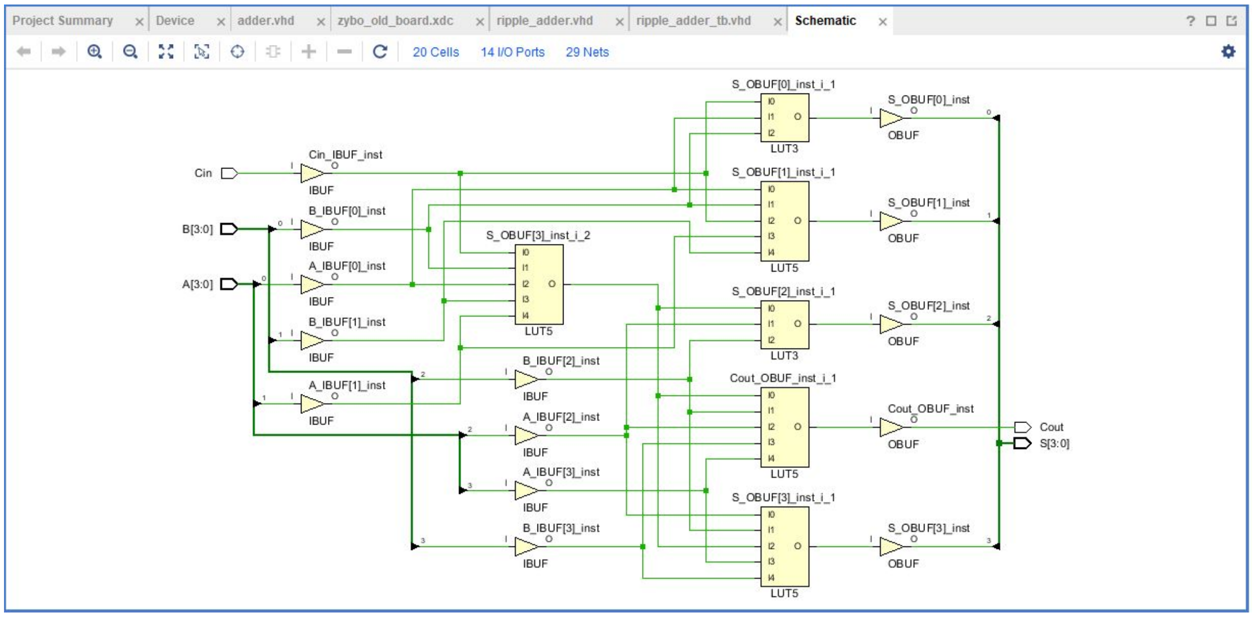
RTL Schematic:



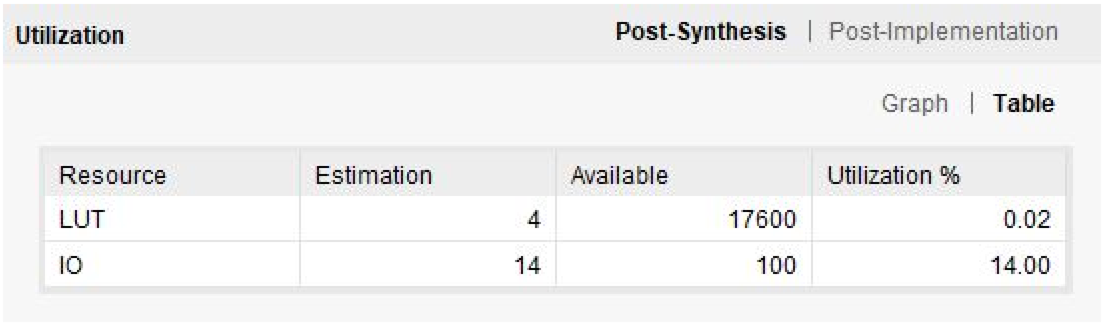
Waveform Simulation:



Synthesis Schematic:

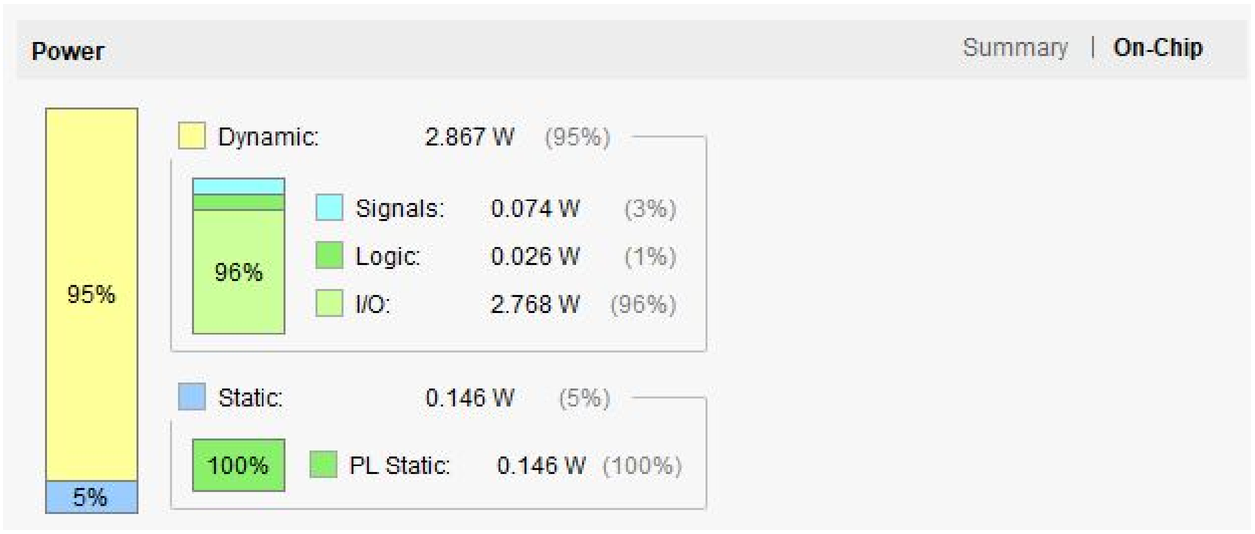


Post Synthesis Utilization Table:



Resource	Estimation	Available	Utilization %
LUT	4	17600	0.02
IO	14	100	14.00

On Chip Power Graph:



🟨	Dynamic:	2.867 W	(95%)
🟦	Signals:	0.074 W	(3%)
🟩	Logic:	0.026 W	(1%)
🟩	I/O:	2.768 W	(96%)
🟦	Static:	0.146 W	(5%)
🟩	PL Static:	0.146 W	(100%)

ALU TESTER

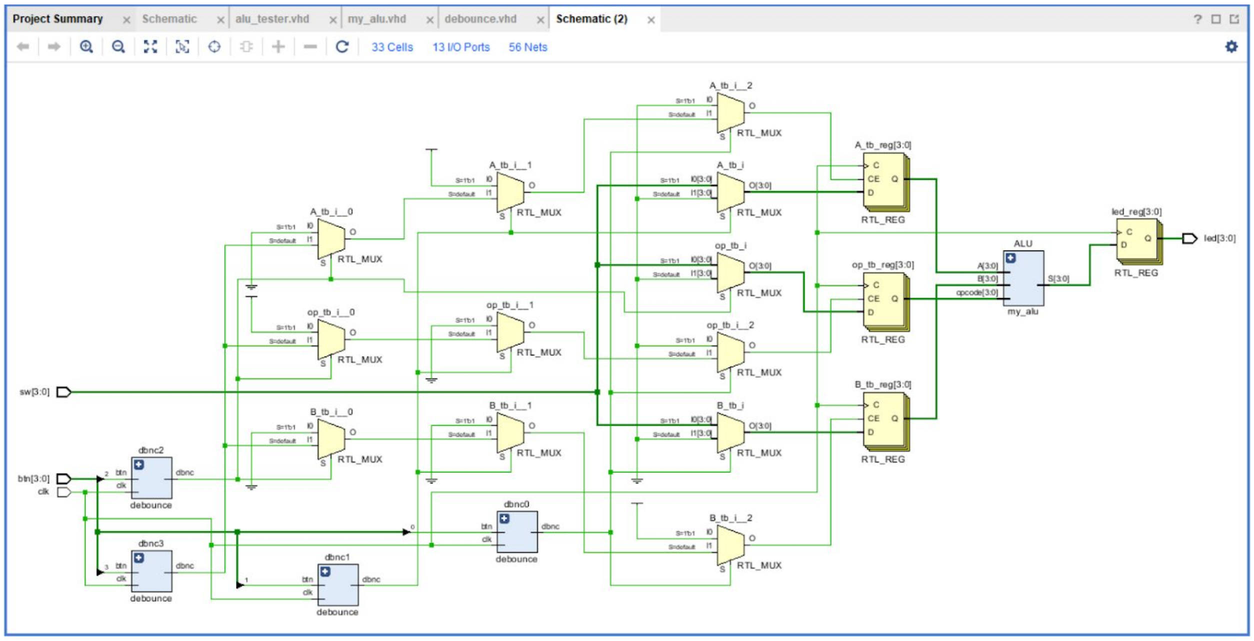
Purpose:

Create a 16 function ALU with the following functions, and debounced buttons.

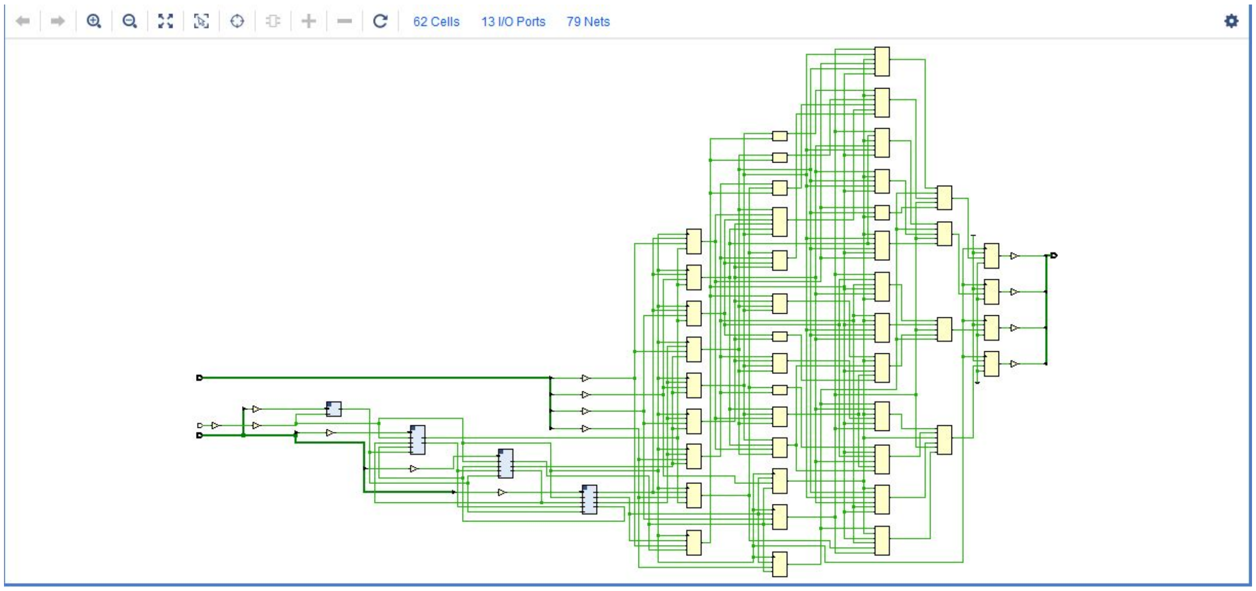
opcode	function	opcode	function
x“0”	$A + B$	x“8”	$A >>> 1$ (right shift arithmetic)
x“1”	$A - B$	x“9”	not A
x“2”	$A + 1$	x“A”	A and B
x“3”	$A - 1$	x“B”	A or B
x“4”	$0 - A$	x“C”	A xor B
x“5”	$A > B$ (greater than - see note)	x“D”	A xnor B
x“6”	$A << 1$ (left shift logical)	x“E”	A nand B
x“7”	$A >> 1$ (right shift logical)	x“F”	A nor B

All Code including, alutester, debounce, my_alu, the bit file, and the constraint file is on github (trp87-rutgers)

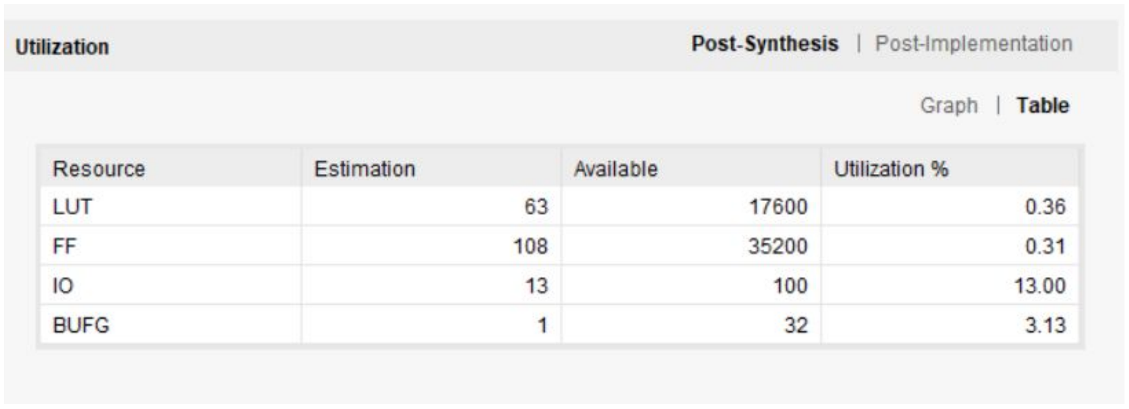
RTL Schematic:



Synthesis Schematic:



Post Synthesis Utilization Table:



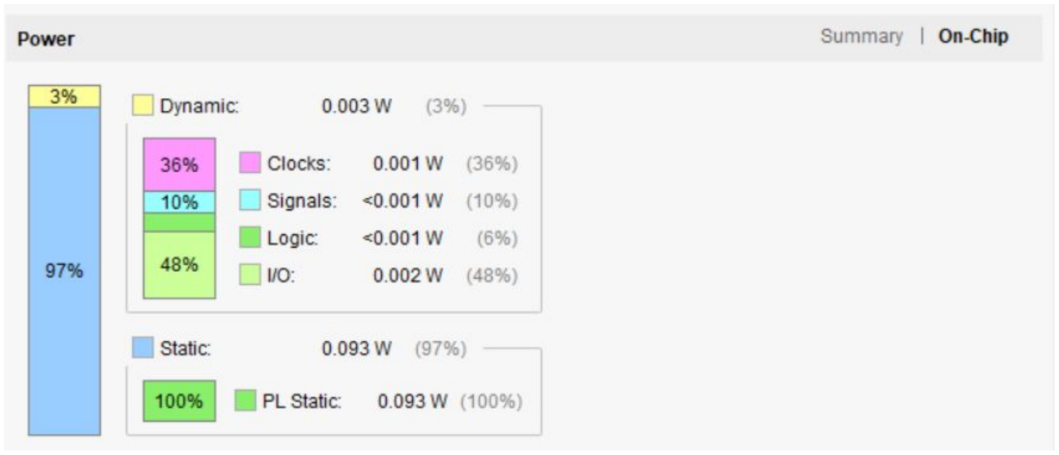
Utilization

Post-Synthesis | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	63	17600	0.36
FF	108	35200	0.31
IO	13	100	13.00
BUFG	1	32	3.13

On Chip Power Graph:



Power

Summary | **On-Chip**

Dynamic: 0.003 W (3%)

Clocks:	0.001 W	(36%)
Signals:	<0.001 W	(10%)
Logic:	<0.001 W	(6%)
I/O:	0.002 W	(48%)

Static: 0.093 W (97%)

PL Static:	0.093 W	(100%)

I learned how to implement a 16 function alu, and the importance of the type a number is. Especially the difference in unsigned and signed numbers while doing operations on them.