**Department of Electrical and Computer Engineering**

Homework Assignment:

**MIPS Assignment #2**

submitted to:

Professor Son Nguyen

ECE 4612: Advanced Processor Systems

Temple University

College of Engineering

1947 North 12th Street

Philadelphia, Pennsylvania 19122

October 23, 2024

prepared by:

Leo Berman  
Email: leo.berman@temple.edu

# Objectives

The objective of this project is to build a processor by building submodules. This includes an adder, subtractor, multiplier, divider, logical unit, shifter, and sign extension. This week I focused on building the adder for the ALU. In order to do this, we have to build up from a half adder to a 64-bit adder.

# Tools/EQUIPMENT

The only tool used for this assignment is Vivado. More specifically, we are programming a Vivado RTL project in Verilog.

# PROCEDURE

Half Adder

1. Exclusive or the two input bits
2. the two input bits with the carry bit

Full Adder

1. Call the Half Adder on the two input bits
2. Call the Half Adder on the output of the previous Half Adder and the input carry bit
3. Or the two output carry bits from the previous operations for the carry out bit

Four Bit Adder

1. Call the Full Adder on the pair of right most bits
2. Take the carry out bit from the previous operation as the carry in bit and call the Full Adder on the next right most pair of bits
3. Repeat Step Two Two More Times

Sixteen Bit Adder

1. Call the Four Bit Adder on the pair of four right most bits
2. Take the carry out bit from the previous operation as the carry in bit and call the Four Bit Adder on the next pair of 4 right most bits
3. Repeat Step two Two more times

Thirty Two Bit Adder

1. Call the Sixteen Bit Adder on the pair of sixteen right most bits
2. Take the carry out bit from the previous operation as the carry in bit and call the Four Bit Adder on the next pair of sixteen right most bits

Sixty Four Bit Adder

1. Call the Thirty Two Bit Adder on the pair of thirty two right most bits
2. Take the carry out bit form the pervious operation as the carry in bit and call the Four Bit Adder on the next pair of thirty two right most bits

# Test Plan

Half Adder

|  |
| --- |
| module t\_halfAdder();  reg a,b;  wire sum,c\_out;  halfAdder AH (.sum(sum), .c\_out(c\_out), .a(a), .b(b));  initial begin  #10 a = 0; b = 0;  #10 a = 0; b = 1;  #10 a = 1; b = 0;  #10 a = 1; b = 1;  #10 $finish;  end  endmodule |

Full Adder

|  |
| --- |
| module t\_fullAdder();  reg a, b, c\_in;  wire sum, c\_out;  fullAdder F1 (.sum(sum),.c\_out(c\_cout), .a(a), .b(b), .c\_in(c\_in));  initial begin  #10 a = 0; b = 0; c\_in = 0;  #10 a = 0; b = 0; c\_in = 1;  #10 a = 0; b = 1; c\_in = 0;  #10 a = 0; b = 1; c\_in = 1;  #10 a = 1; b = 0; c\_in = 0;  #10 a = 1; b = 0; c\_in = 1;  #10 a = 1; b = 1; c\_in = 0;  #10 a = 1; b = 1; c\_in = 1;  #10 $finish ;  end  endmodule |

Four Bit Adder

|  |
| --- |
| module t\_fourBitAdder();  reg[3:0] a,b;  reg c\_in;  wire[3:0] sum;  wire c\_out;  fourBitAdder tF1(.sum(sum), .c\_out(c\_out), .a(a), .b(b), .c\_in(c\_in));  initial begin  #10 a = 0; b = 0; c\_in = 0;  #10 a = 7; b = 7; c\_in = 1;  #10 a = 7; b = 8; c\_in = 1;  #10 a = 5; b = 5; c\_in = 1;  #10 $finish;  end  endmodule |

Sixteen Bit Adder

|  |
| --- |
| module t\_sixteenBitAdder();  reg[15:0] a,b;  reg c\_in;  wire[15:0] sum;  wire c\_out;  sixteenBitAdder tF1(.sum(sum), .c\_out(c\_out), .a(a), .b(b), .c\_in(c\_in));  initial begin  #10 a = 65534; b = 1; c\_in = 0;  #10 a = 32767; b = 32767; c\_in = 0;  #10 a = 30000; b = 8; c\_in = 1;  #10 a = 25; b = 1000; c\_in = 1;  #10 $finish;  end  endmodule |

Thirty Two Bit Adder

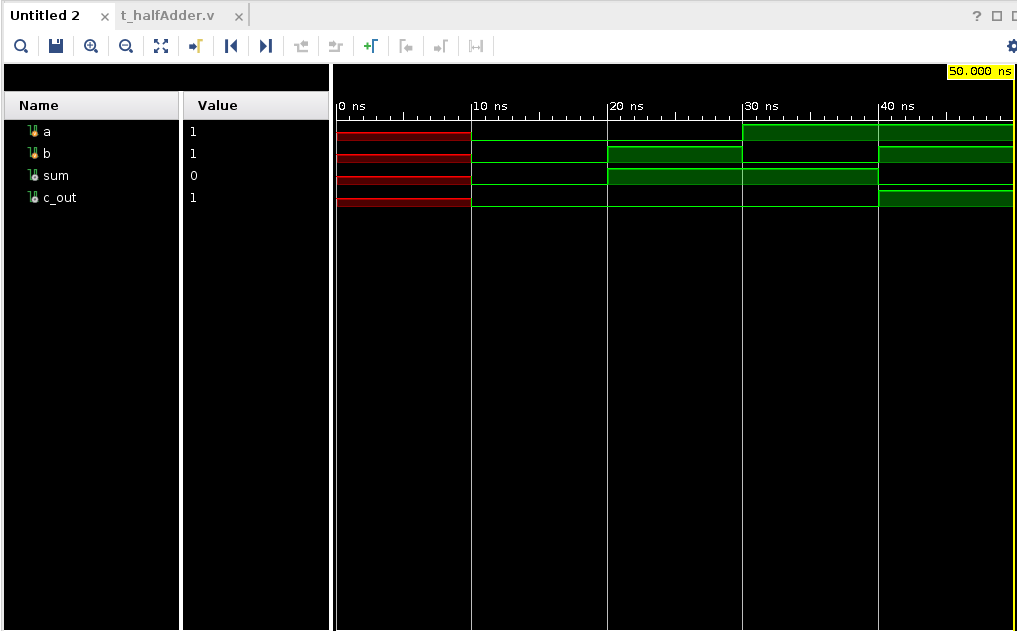
|  |
| --- |
| module t\_thirtytwoBitAdder();  reg[31:0] a,b;  reg c\_in;  wire[31:0] sum;  wire c\_out;  thirtytwoBitAdder tF1(.sum(sum), .c\_out(c\_out), .a(a), .b(b), .c\_in(c\_in));  initial begin  #10 a = 4294967294; b = 1; c\_in = 0;  #10 a = 2147483647; b = 2147483648; c\_in = 0;  #10 a = 300000; b = 8; c\_in = 1;  #10 a = 25; b = 1000; c\_in = 1;  #10 $finish;  end  endmodule |

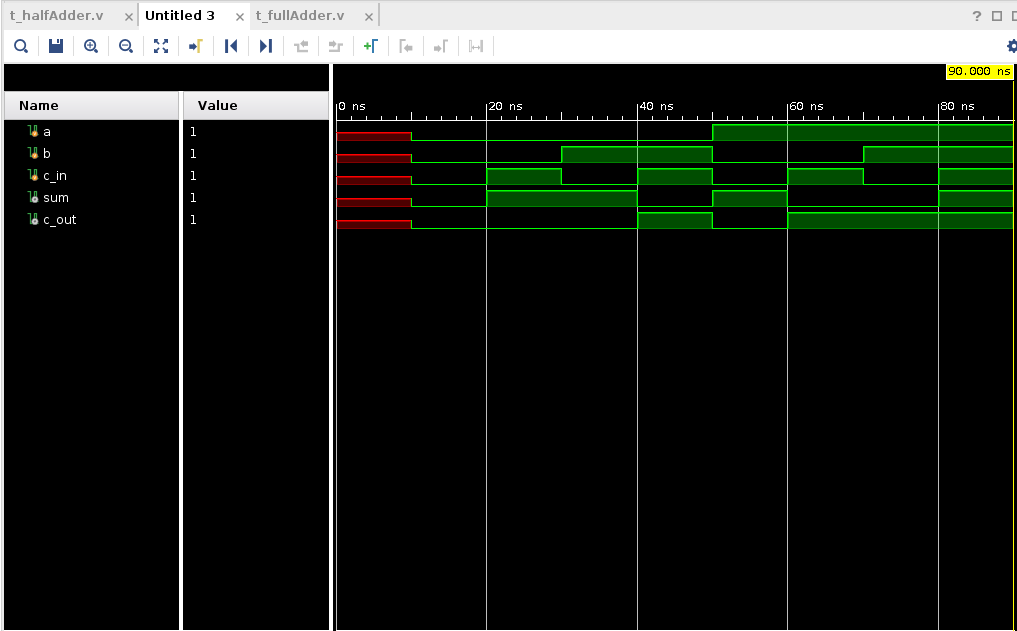
Sixty Four Bit Adder

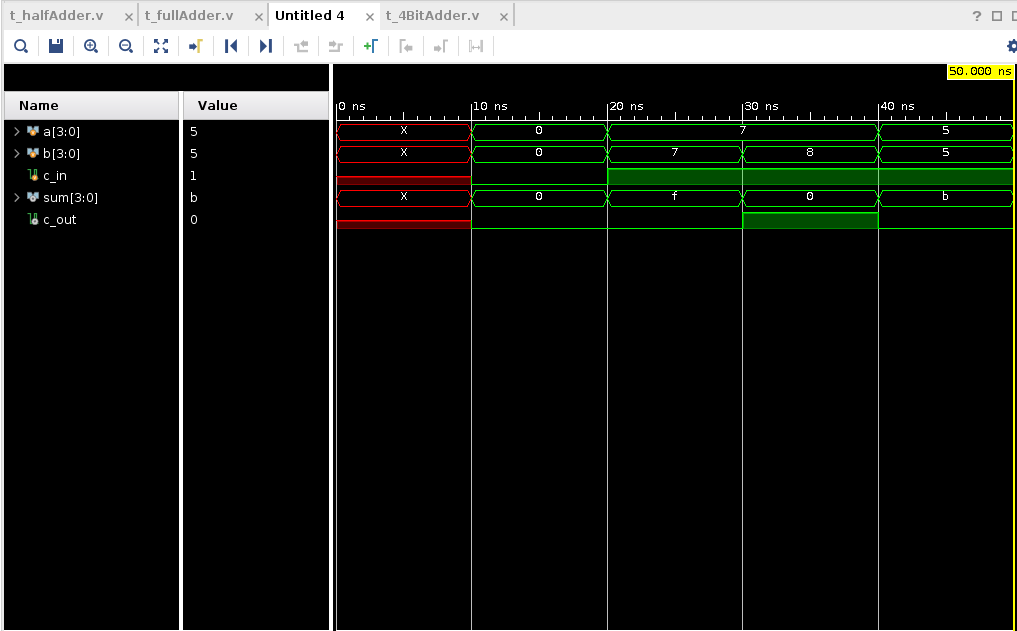
|  |
| --- |
| module t\_sixtyfourBitAdder();  reg[63:0] a,b;  reg c\_in;  wire[63:0] sum;  wire c\_out;  sixtyfourBitAdder tF1(.sum(sum), .c\_out(c\_out), .a(a), .b(b), .c\_in(c\_in));  initial begin  #10 a = 18446744073709551614; b = 1; c\_in = 0;  #10 a = 9223372036854775807; b = 9223372036854775808; c\_in = 1;  #10 a = 300000; b = 8; c\_in = 1;  #10 a = 25; b = 1000; c\_in = 1;  #10 $finish;  end  endmodule |

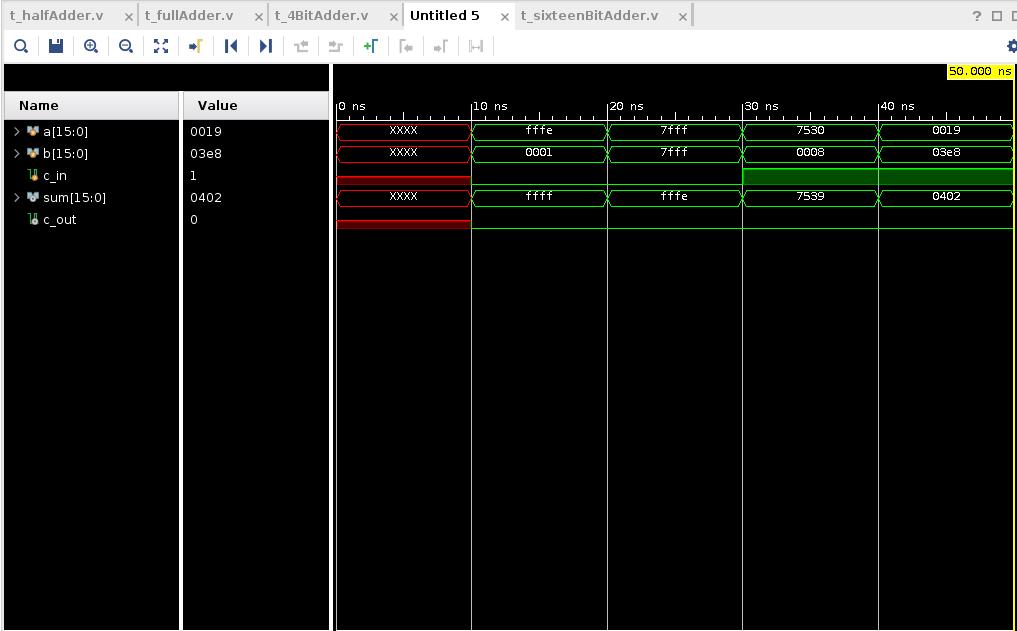
# RESULTS

Half Adder

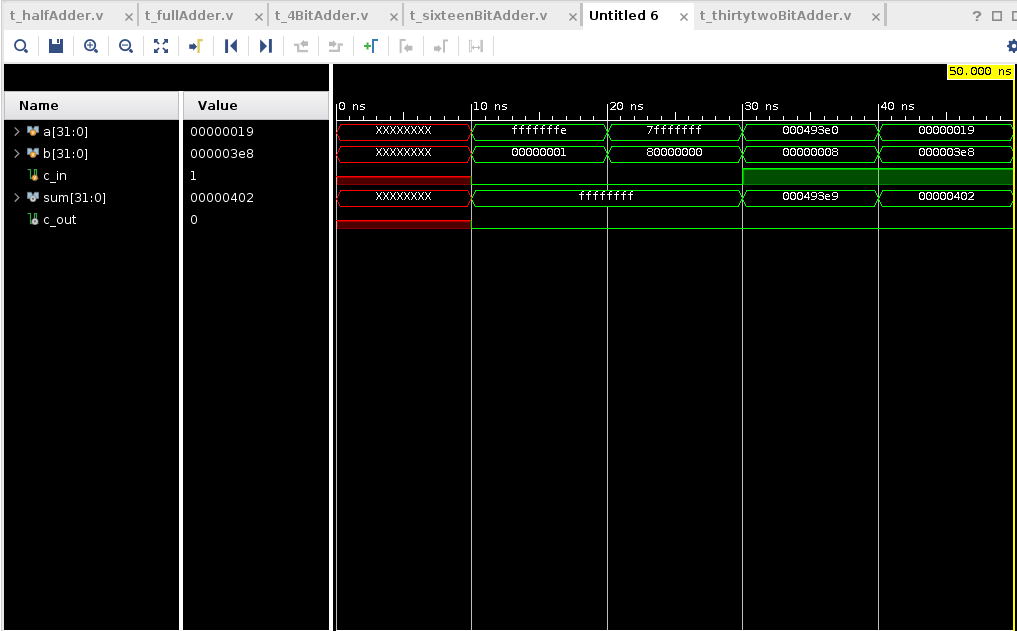
Full Adder

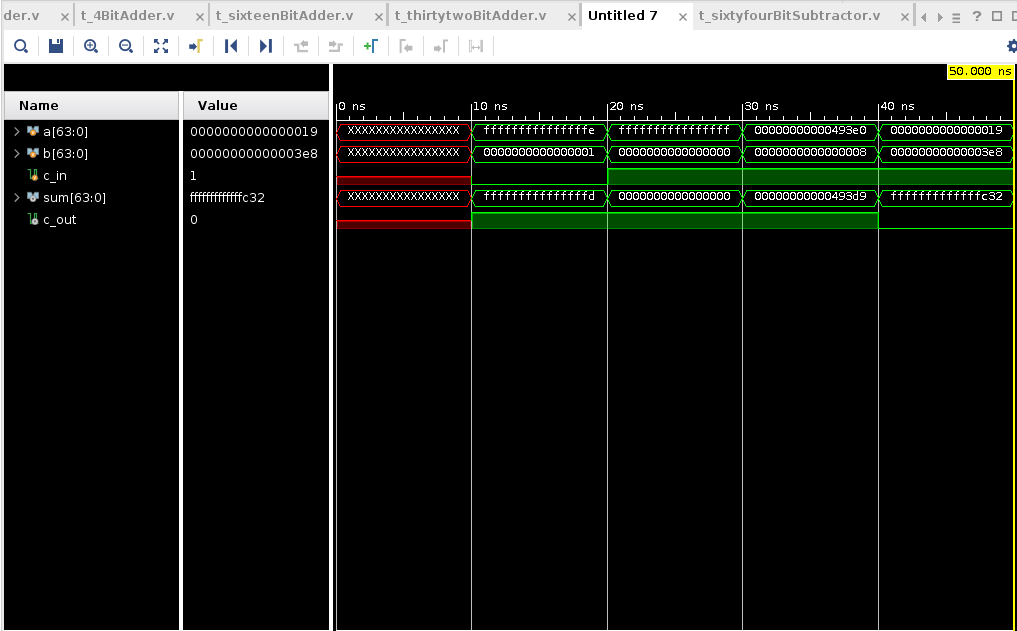
  
  
Four Bit Adder

Sixteen Bit Adder



Thirty Two Bit Adder

Sixty Four Bit Adder



# Conclusion

This assignment was overall relatively straight forward. Since the beginning primitive examples were given to us scaling up was relatively easy. I don’t feel like I came up with the most effective test cases, but my train of thought was to utilize the maximum register size to ensure that I was using the variables in the correct order and then keeping relatively similar test cases in common just to have a check and see if there was anything glaringly wrong.