# ECSE 318 Lab 1 Report

## Group 5

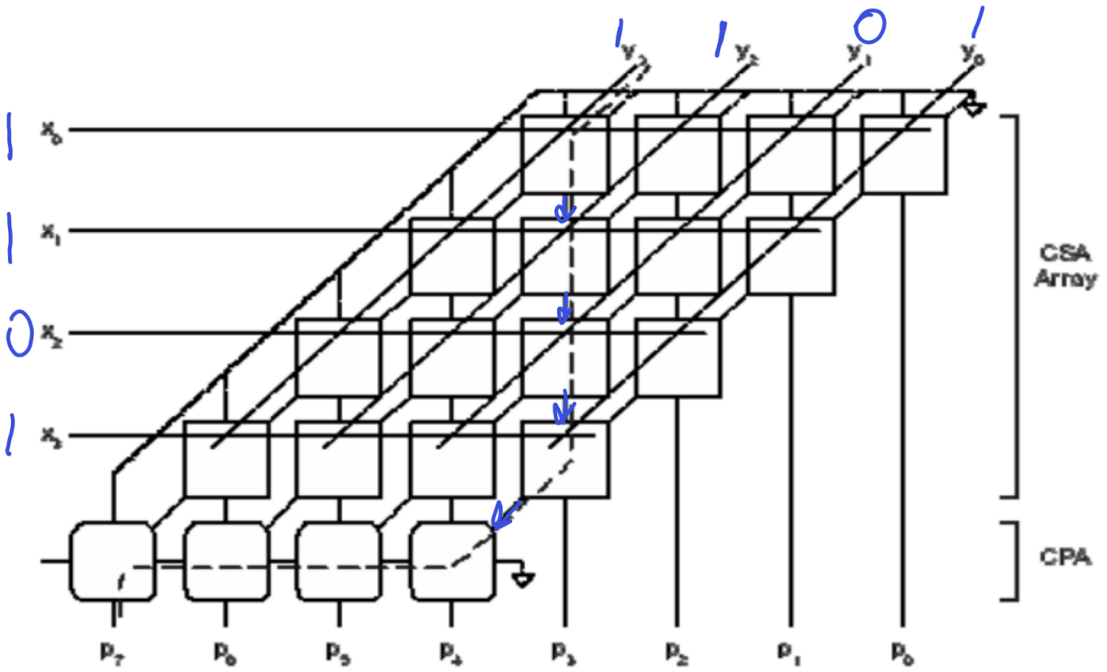
Nabeeh Daouk and Kyle Heston

Tuesday, September 24, 2023

**Problem 1**

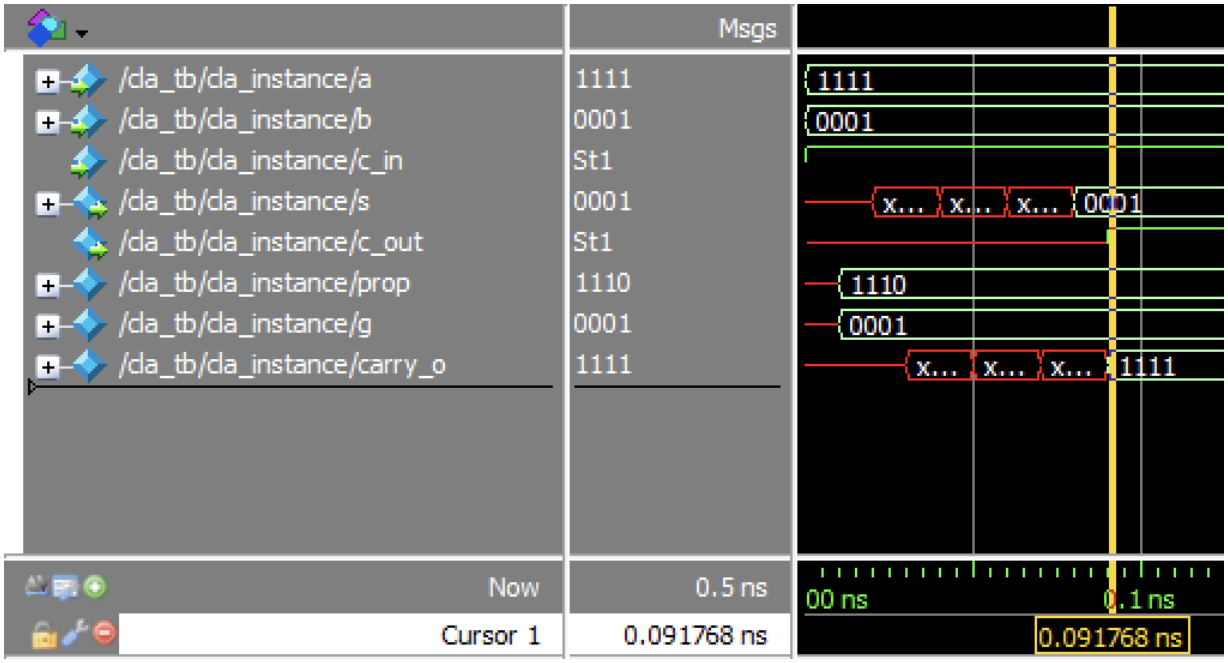
```
Testing Multiplication Functionality for 2*6 Below:
Multiplicand: 0010 *  Multiplier:0110   =  Product:00001100
Multiplicand:  2 *  Multiplier: 6   =  Product: 12

Testing Multiplication Functionality for 12*3 Below:
Multiplicand: 1100 *  Multiplier:0011   =  Product:00100100
Multiplicand: 12 *  Multiplier: 3   =  Product: 36

ACTIVATING DASHED PATH, Question 1c:
Testing Multiplication Functionality for 12*3 Below:
Multiplicand: 1101 *  Multiplier:1101   =  Product:10101001
Multiplicand: 13 *  Multiplier:13   =  Product:169
```

*Problem 1 Testbench Results*




***Critical Path Input:*** input sequence of x=1101 and y=1101 activates the critical path

**Problem 2**



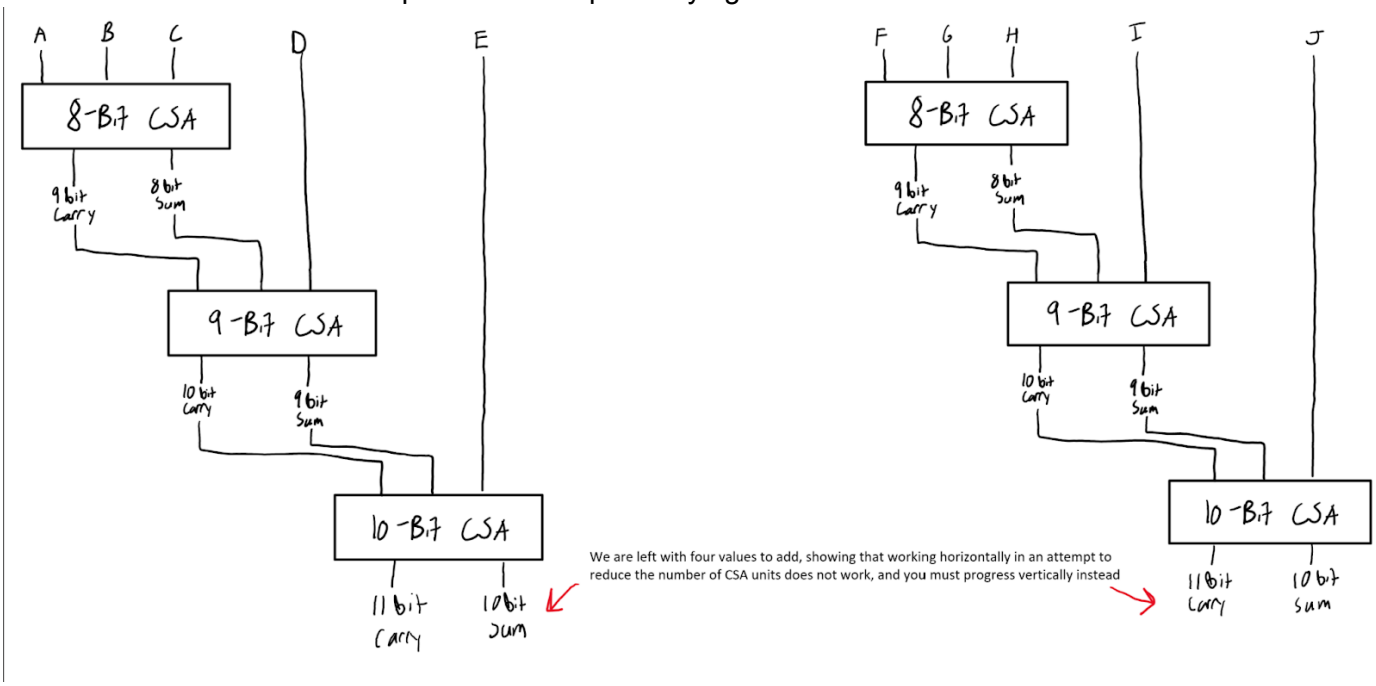

#10 delay in ps; 0.91 ns = ~90ps; 9 Gate Delay to c_out, 8 to sum

```
LONGEST PROPIGATION DELAY:
A: 1111 +  B:0001   (c_in=1) =  Sum:0001   (c_out=1)
A: 15 +  B: 1   (c_in=1) =  Sum: 1   (c_out=1)

Testing Addition Functionality:
A: 0101 +  B:0111   (c_in=0) =  Sum:1100   (c_out=0)
A:  5 +  B: 7   (c_in=0) =  Sum:12   (c_out=0)

Testing Addition Functionality:
A: 0100 +  B:0011   (c_in=1) =  Sum:1000   (c_out=0)
A:  4 +  B: 3   (c_in=1) =  Sum: 8   (c_out=0)

Testing Addition Functionality:
A: 1001 +  B:0101   (c_in=1) =  Sum:1111   (c_out=0)
A:  9 +  B: 5   (c_in=1) =  Sum:15   (c_out=0)
```

*Problem 2 Testbench Results*

## Problem 3

A) Eight CSA stages are needed. The first CSA stage takes three inputs, 10-3=7, we need 7 additional stages to take in all the inputs. An example of trying to use less than 8 CSA units is shown here:

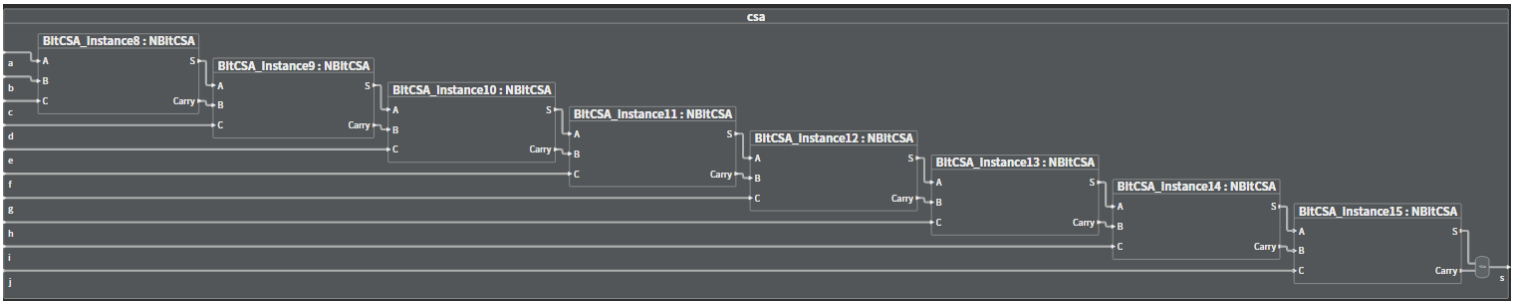



**We cannot simplify the four values using a full adder, which takes three inputs.**
**8 Total stages are used in Verilog design**

```
# run -all
# Test addition problem 1:
# A: 00001011 +  B:00000010 + C:00001101 + D:00000100 + E:00000101 + F:00000110 + G:00000111 + H:00001000 + I:00001001 + J:00001010 = Sum:0000000000001001011
# A:  11 +  B:  2 + C: 13 + D:  4 + E:  5 + F:  6 + G:  7 + H:  8 + I:  9 + J: 10 = Sum:    75
#
# Test addition problem 2:
# A: 00000011 +  B:00001110 + C:00000101 + D:00000110 + E:00000111 + F:00001000 + G:00010011 + H:00001010 + I:00000000 + J:00000000 = Sum:0000000000001001000
# A:   3 +  B: 14 + C:  5 + D:  6 + E:  7 + F:  8 + G: 19 + H: 10 + I:  0 + J:  0 = Sum:    72
#
# Test additional example:
# A: 00011110 +  B:00010110 + C:00010011 + D:01111110 + E:00000101 + F:01011100 + G:01000101 + H:00101100 + I:00000001 + J:00001010 = Sum:0000000000110100010
# A:  30 +  B: 22 + C: 19 + D:126 + E:  5 + F: 92 + G: 69 + H: 44 + I:  1 + J: 10 = Sum:   418
#
# Test additional example:
# A: 11111111 +  B:11111111 + C:11111111 + D:11111111 + E:11111111 + F:11111111 + G:11111111 + H:11111111 + I:11111111 + J:11111110 = Sum:0000000100111110101
# A: 255 +  B:255 + C:255 + D:255 + E:255 + F:255 + G:255 + H:255 + I:255 + J:254 = Sum:  2549
#  quit -f
# End time: 23:02:27 on Sep 22,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
```



*Problem 3 Testbench Results*




*Problem 3 CSA BLock Diagram*

**Problem 4**

```
# Test multiplication problem 1:
# Multiplicand: 10110 *  Multiplier:00100 = Product:1111011000
# Multiplicand: -10 *  Multiplier:  4 = Product:   -40
#
# Test multiplication problem 2:
# Multiplicand: 01011 *  Multiplier:11101 = Product:1111011111
# Multiplicand:  11 *  Multiplier:  -3 = Product:   -33
#
# Test multiplication problem 3:
# Multiplicand: 10110 *  Multiplier:10101 = Product:0001101110
# Multiplicand: -10 *  Multiplier: -11 = Product:   110
```

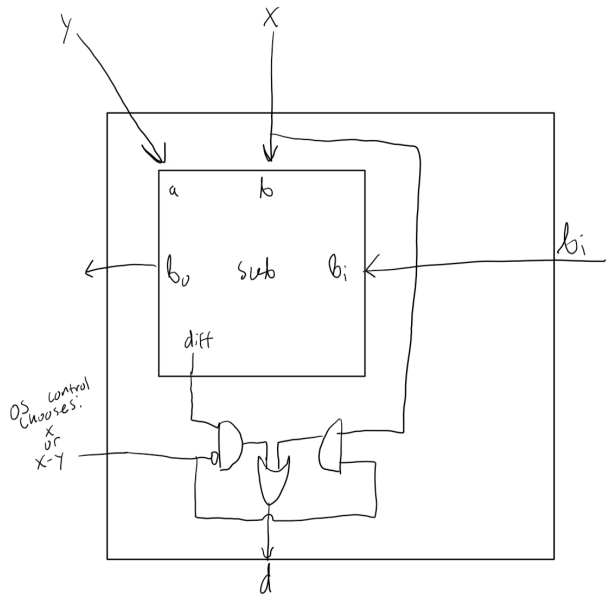
*Problem 4 Testbench Results*

**Problem 5**

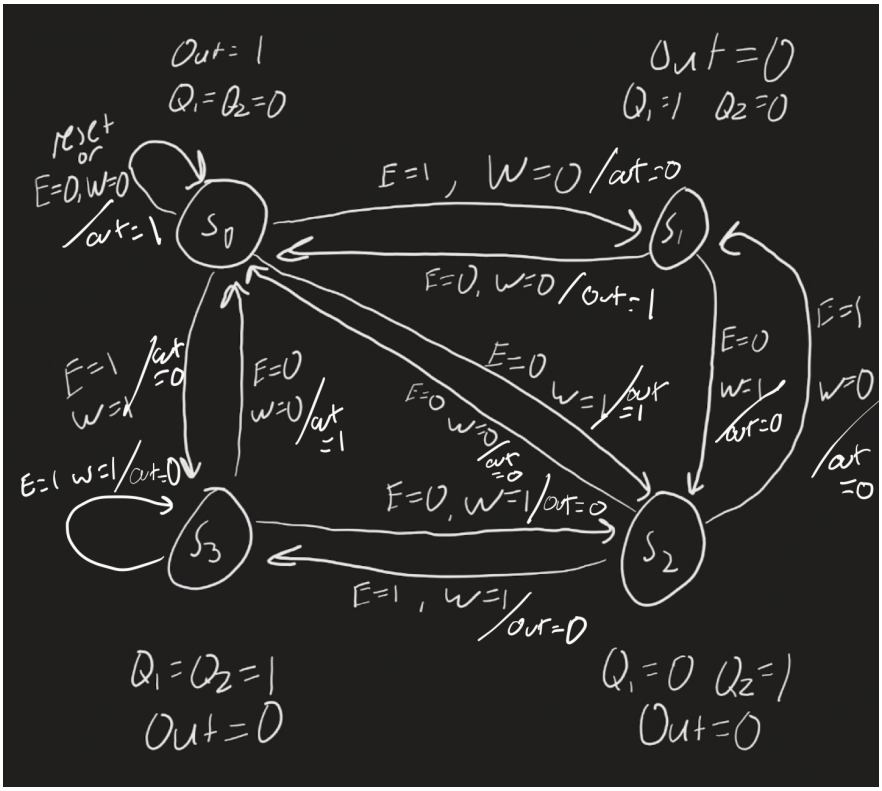
```
# Testing Division Functionality for 6/2 Below:
# Dividend: 0110 *  Divisor: 0010   =  Quotient:0011 and Remainder: 0000
# Dividend:  6 *  Divisor:  2   =  Quotient: 3 and Remainder:  0
#
# Testing Division Functionality for 7/2 Below:
# Dividend: 0111 *  Divisor: 0010   =  Quotient: 0011 and Remainder:  1
# Dividend:  7 *  Divisor:  2   =  Quotient:  3 and Remainder:  1
#  quit -f
# End time: 01:14:26 on Sep 22,2023, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
```

*Problem 5 Testbench Results*

Created a parallel divider using an ARRAY of subtractor instances as drawn below--x is dividend, y is divisor. Array is described in Verilog.

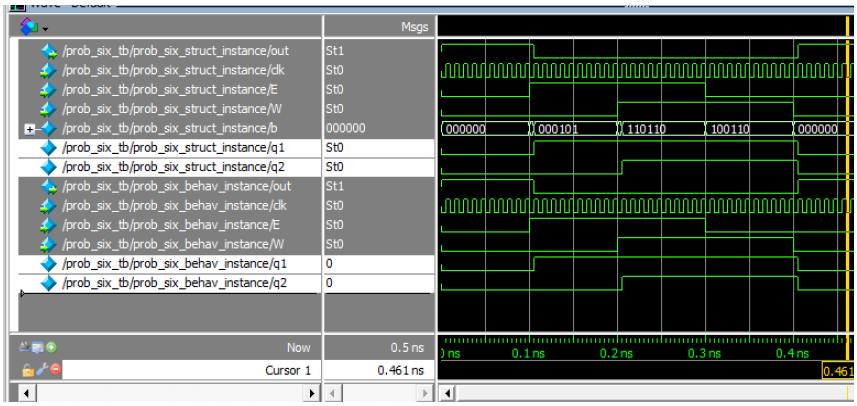


## Problem 6




*Problem 6 State Diagram*

```
STATE=q2q1

Start in S0::
E: 0       W:0       BEHAVIORAL_OUT:1       STRUCTURAL_OUT:1

Transition to S1
E: 1       W:0       BEHAVIORAL_OUT:0       STRUCTURAL_OUT:0

Transition to S2
E: 1       W:1       BEHAVIORAL_OUT:0       STRUCTURAL_OUT:0

Transition to S3
E: 0       W:1       BEHAVIORAL_OUT:0       STRUCTURAL_OUT:0

Transition Back to S0
E: 0       W:0       BEHAVIORAL_OUT:1       STRUCTURAL_OUT:1
```




*Problem 6 Testbench Results*

B) The state diagram drawn shows what Out equals based on input E and W. The Verilog models show the same input/output combinations as the drawn diagram. Follows expected state transitions and outputs (seen in plots (state value registers highlighted above in waveform). We can see that the structural and behavioral implementations have the same states.

D) Both behavioral and structural models have the same testbench output.