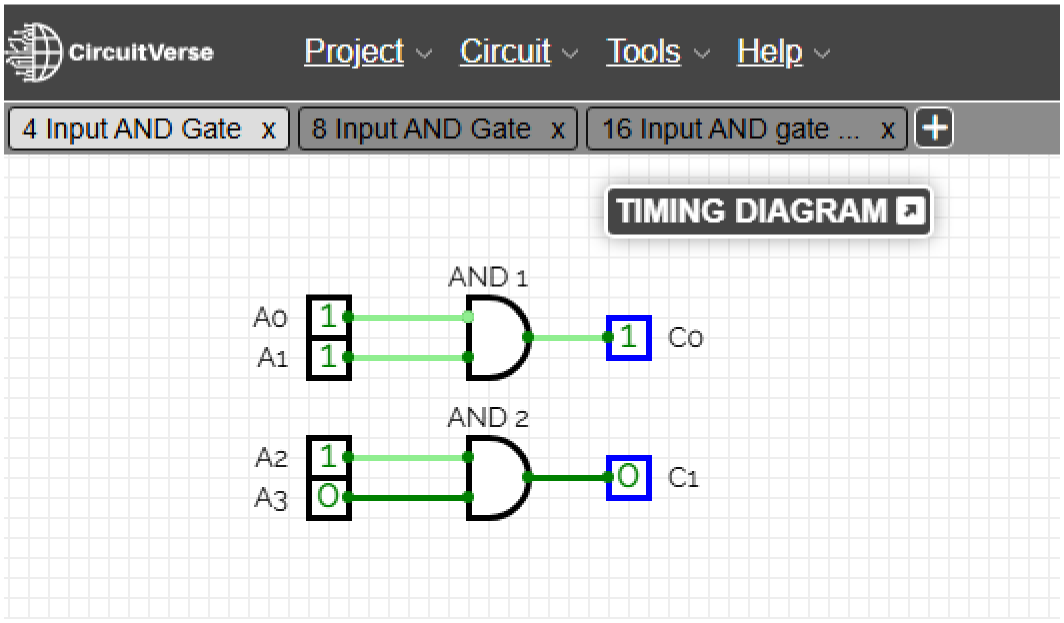
Final Project: 8-Bit Binary Calculalor

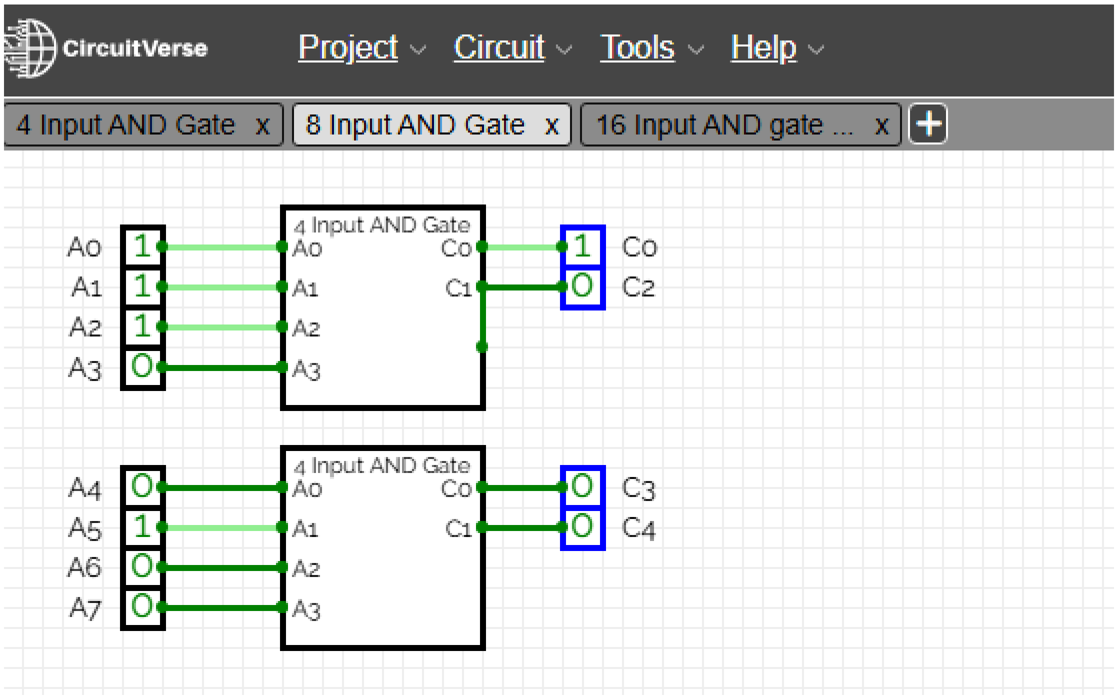
By: Josue Palacios

<u>Setup and Installation Confirmed</u>

For this project, I will be utilizing CircuitVerse as it requires no installation or java compatibility. The setup was very simple as I only needed to create an account with no installation necessary. To familiarize myself with CircuitVerse, I began by watching tutorial videos created by DIGITEK KEYS on YouTube, where I learned how to construct simple AND gates. Through these tutorials, I discovered how to build more complex circuits by utilizing subcircuits. For example, I transformed a 4-input AND gate into an 8-input AND circuit, which allowed for the demonstration of all possible input combinations for an AND logic operation. This exercise was very insightful on creating inputs, outputs, gates, labels, and subcircuits.



In this second image, I used the previous circuit as a subcircuit to create a larger circuit capable of having 8 inputs.



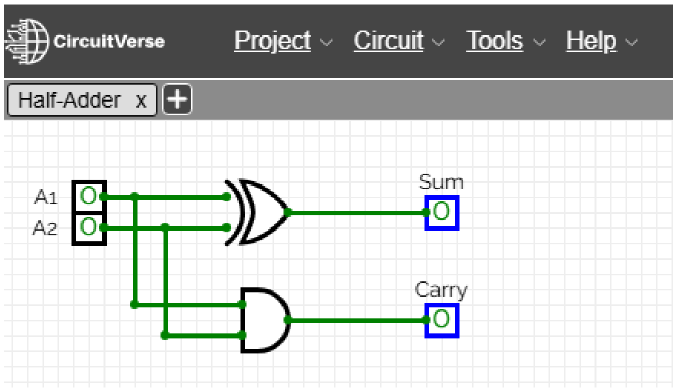
Build and test an 8-bit adder

To begin constructing the 8-bit adder, I began by designing a half adder, which served as the foundational component for developing a full adder. During this phase, I consulted the course textbook to reinforce my understanding of combinational logic, such as the role of Boolean operators, input variables, and their corresponding outputs. This knowledge is essential, as the adder functions as a combinational circuit, where the output is determined solely by the given inputs.

The half adder enables the addition of any two binary digits. Its logical behavior is illustrated through the truth table I created using Excel, presented below.

Inputs		Outputs	
X	Y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The half-adder made in CircuitVerse was tested for every possibility and matched the truth table. I have provided an image from below



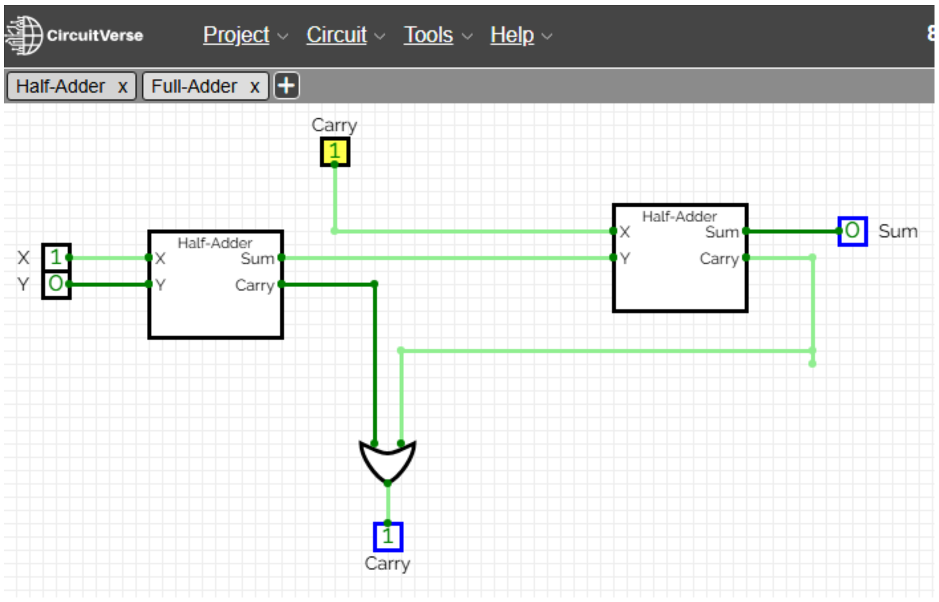
Using the truth table, we can see that the sum column resembles XOR while carry resembles AND which are the gates that make up the circuit

However, the half adder alone is insufficient for performing multi-bit binary addition, as it is limited to adding only two single-bit values. To accommodate larger binary numbers, a full adder circuit must be constructed. The full adder enables the addition of binary numbers across multiple bits by incorporating an additional input to account for the carry bit generated from a previous addition, much like the carry-over process in base-10 arithmetic. This process mirrors traditional decimal addition, beginning with the rightmost column, note the unit's digit, carry the tens digit, add that carry to the next column, and repeat the process however many times is needed. To replicate this in binary, the full adder requires three inputs which will be X, Y, and a Carry which again produces the two outputs sum and carry.

Moving forward, the full-adder needs to be established. The truth table for it is below.

Inputs			Outputs	
X	Y	Carry	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

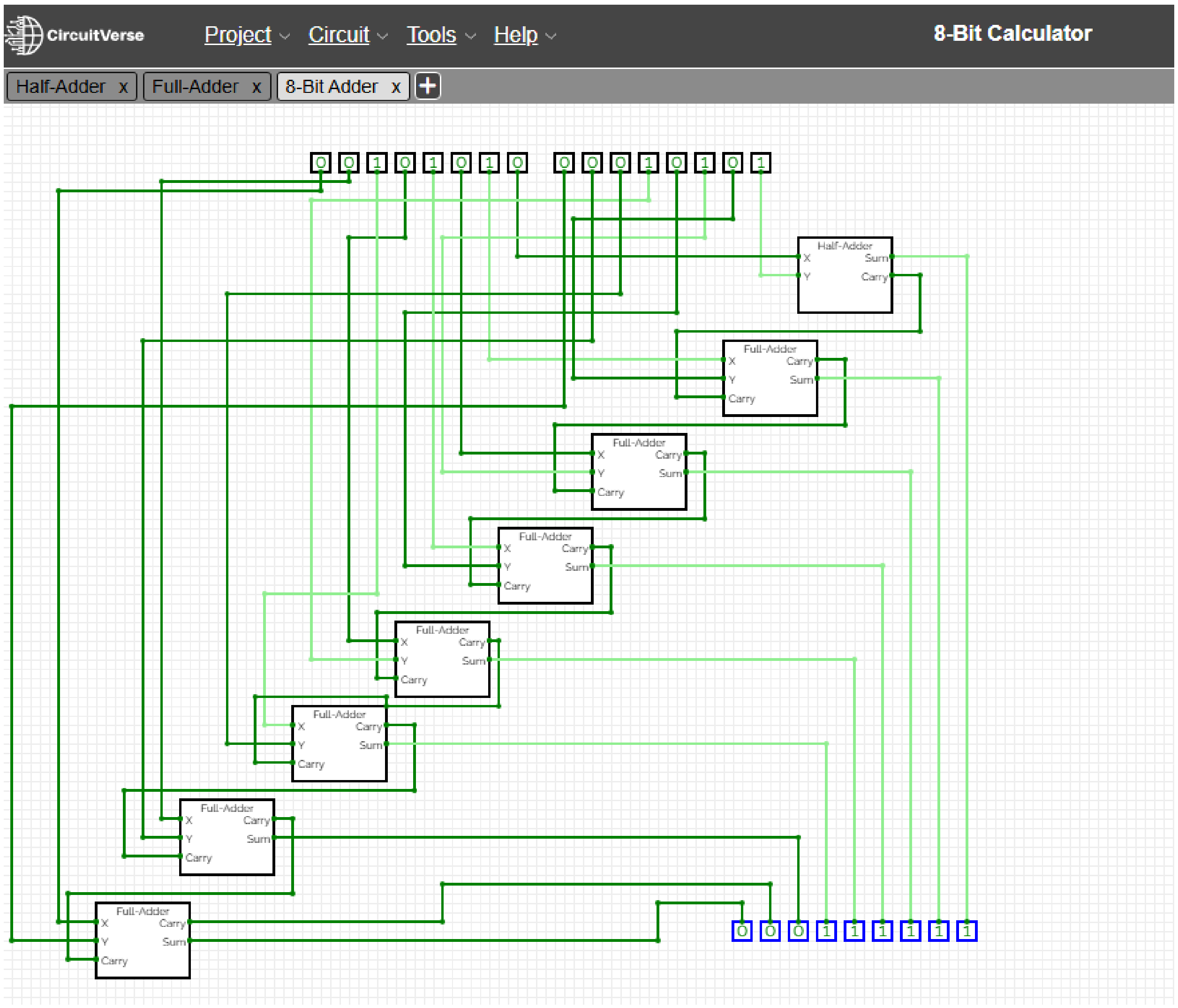
The full-adder made in CircuitVerse was tested for every possibility and matched the truth table. I have provided an image from below



Here, an additional AND gate with two half-adders are used to incorporate the Carry input to complete the objective mentioned before.

With the full-adder made, now the 8-bit adder can be made by using the full-adder. Since the full-adder is only capable of adding three bits, the 8-bit adder will consist of replicating the full-adder circuit 8 times to feed the carry out of a circuit into the carry in of the following circuit which makes it a ripple-carry adder.

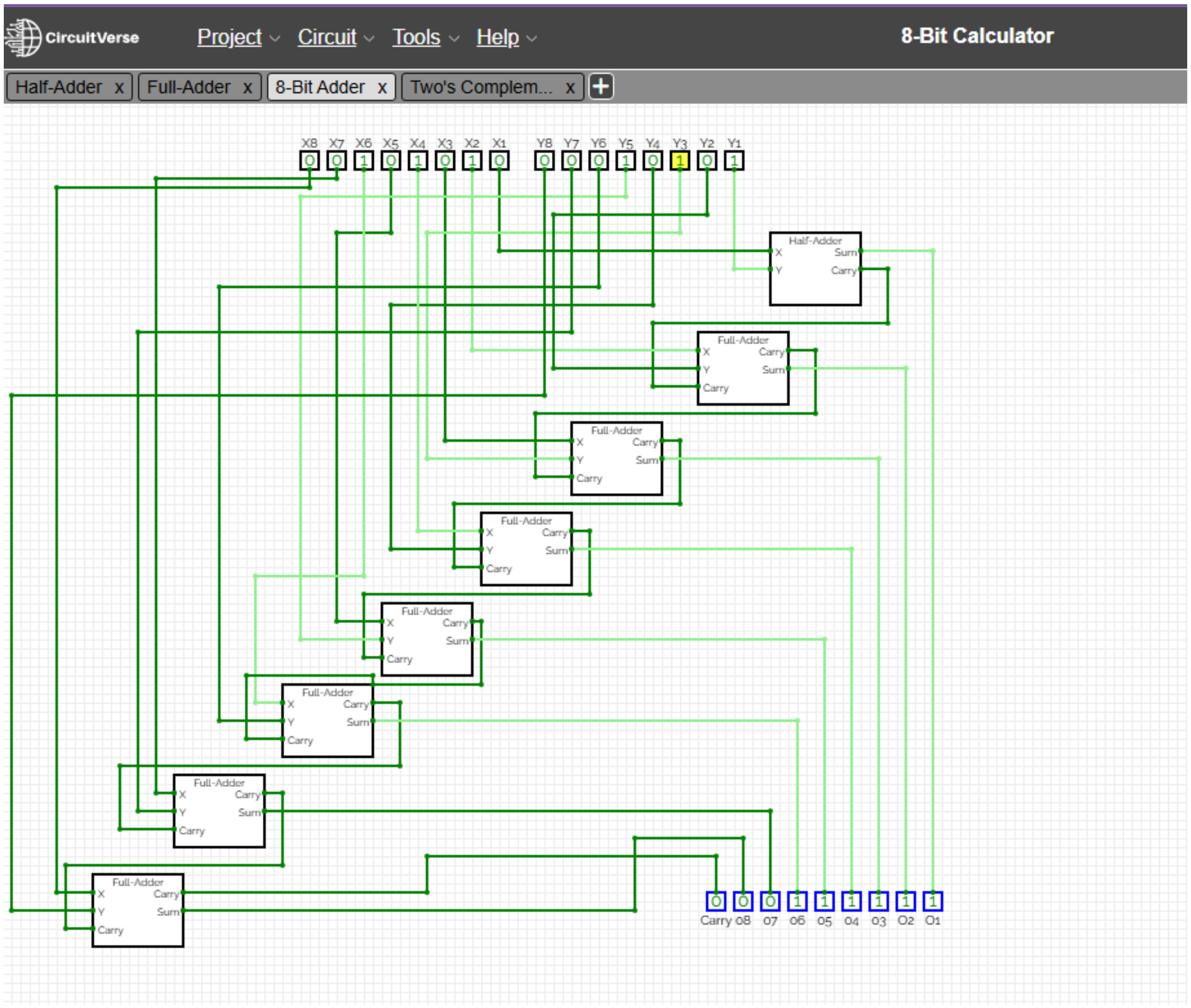
The 8-bit Adder made in CircuitVerse was tested with known binary addition examples (for example 00101010 + 00010101). I have provided an image from below



Notes of Complications: After completing the previous example, I realized that my failure to label the inputs and outputs of the circuit was causing significant confusion, particularly as I attempted to integrate the 8-bit adder into the Two's Complement Module. This lack of labeling ultimately contributed to a misconnection issue where the O8 and carry nodes were inadvertently swapped.

At that point, I was already deep into the construction of the Two's Complement Module and initially believed that correcting the error would require rebuilding the entire subcircuit. However, I was relieved to discover that any modifications made to a subcircuit are automatically reflected wherever the circuit is reused. This dynamic updating feature significantly reduced the workload and prevented the need to recreate the entire module from scratch.

The redone 8-bit Adder can be found below

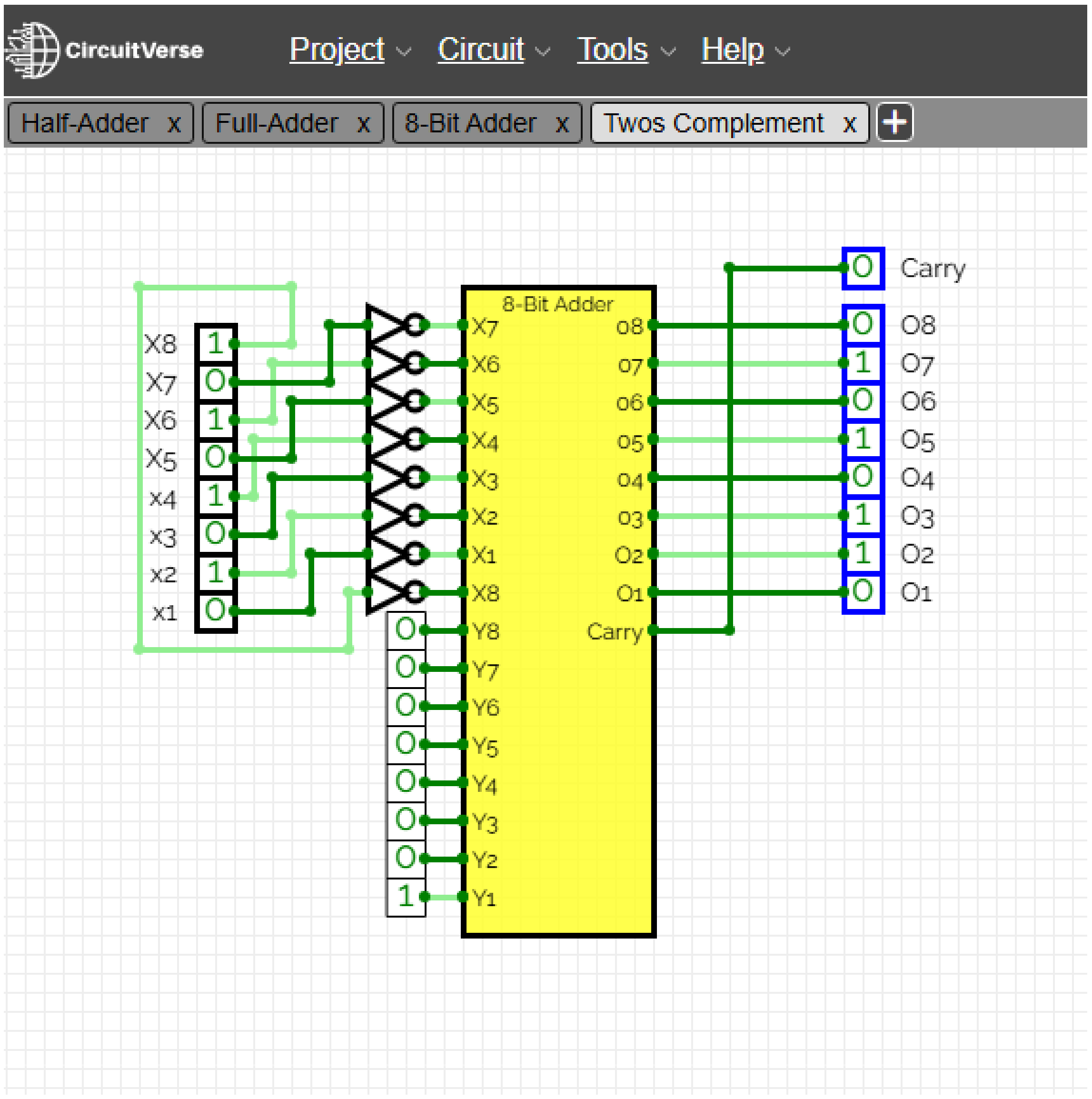


In the known example, 00101010 + 00010101 is 42 + 21 in decimal which is equal to 63. We can see that when entered into my calculator in CircuitVerse 00101010 + 00010101 = 000111111, which is 42 + 21 = 63, matching the examples' input and output.

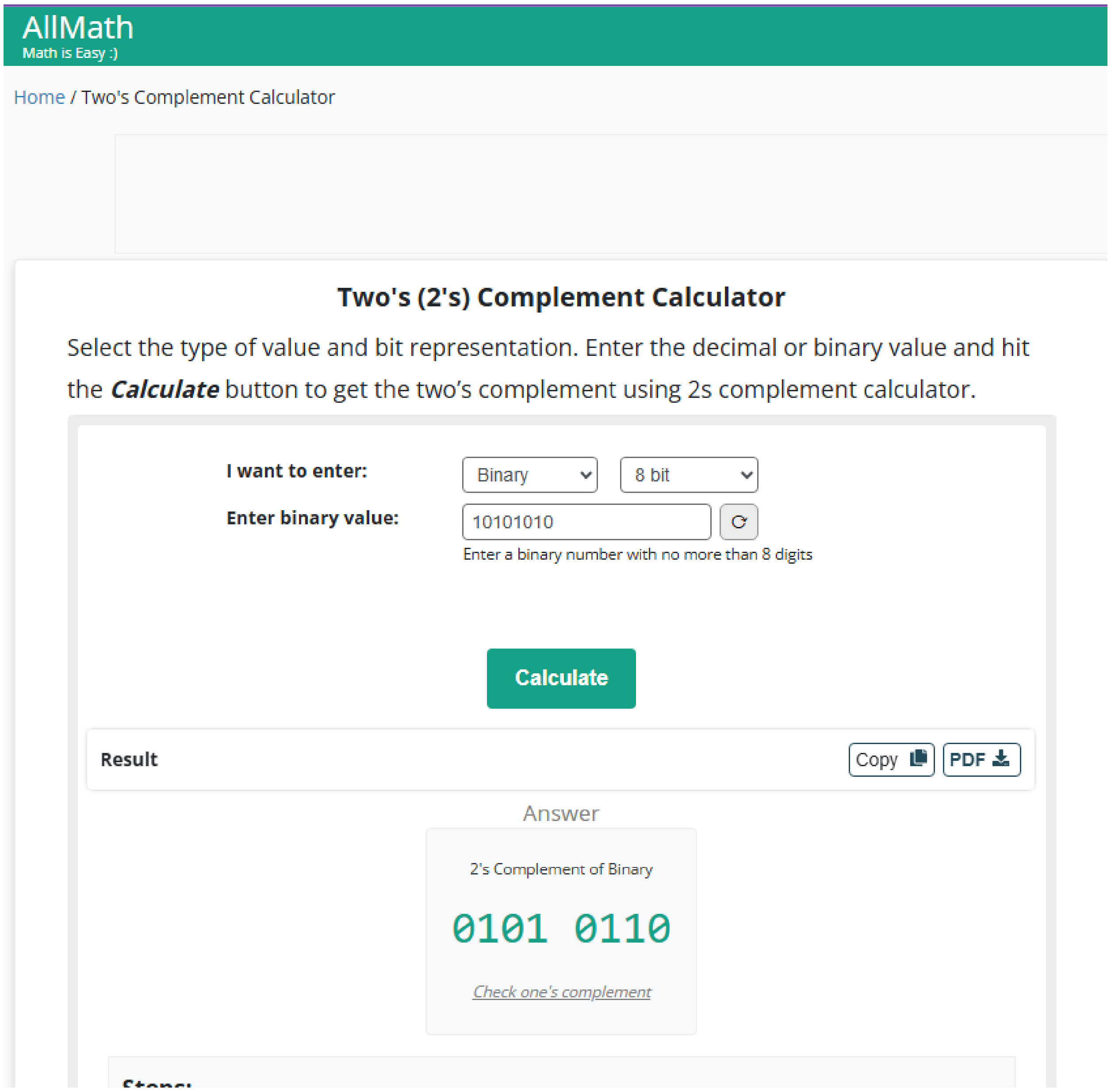
<u>Developing and testing two's complement module</u>

To implement subtraction by using two's complement, I needed to create a module convert it by inverting the subtrahend 8-bit binary number and then add 1. I did this by using the 8-bit adder, not gate, and fixed inputs. The not gate would invert the subtrahend, then the fixed inputs would represent one, followed by the two being added together to make the two's complement of a number.

The module made in CircuitVerse can be seen below



To test this module, I used to separate websites for calculating two's complement. Pictures and citations will be down below.


AllMath
Math is Easy :)

Home / Two's Complement Calculator

Two's (2's) Complement Calculator

Select the type of value and bit representation. Enter the decimal or binary value and hit the ***Calculate*** button to get the two's complement using 2s complement calculator.

I want to enter: Binary 8 bit

Enter binary value: 10101010

Enter a binary number with no more than 8 digits

Calculate

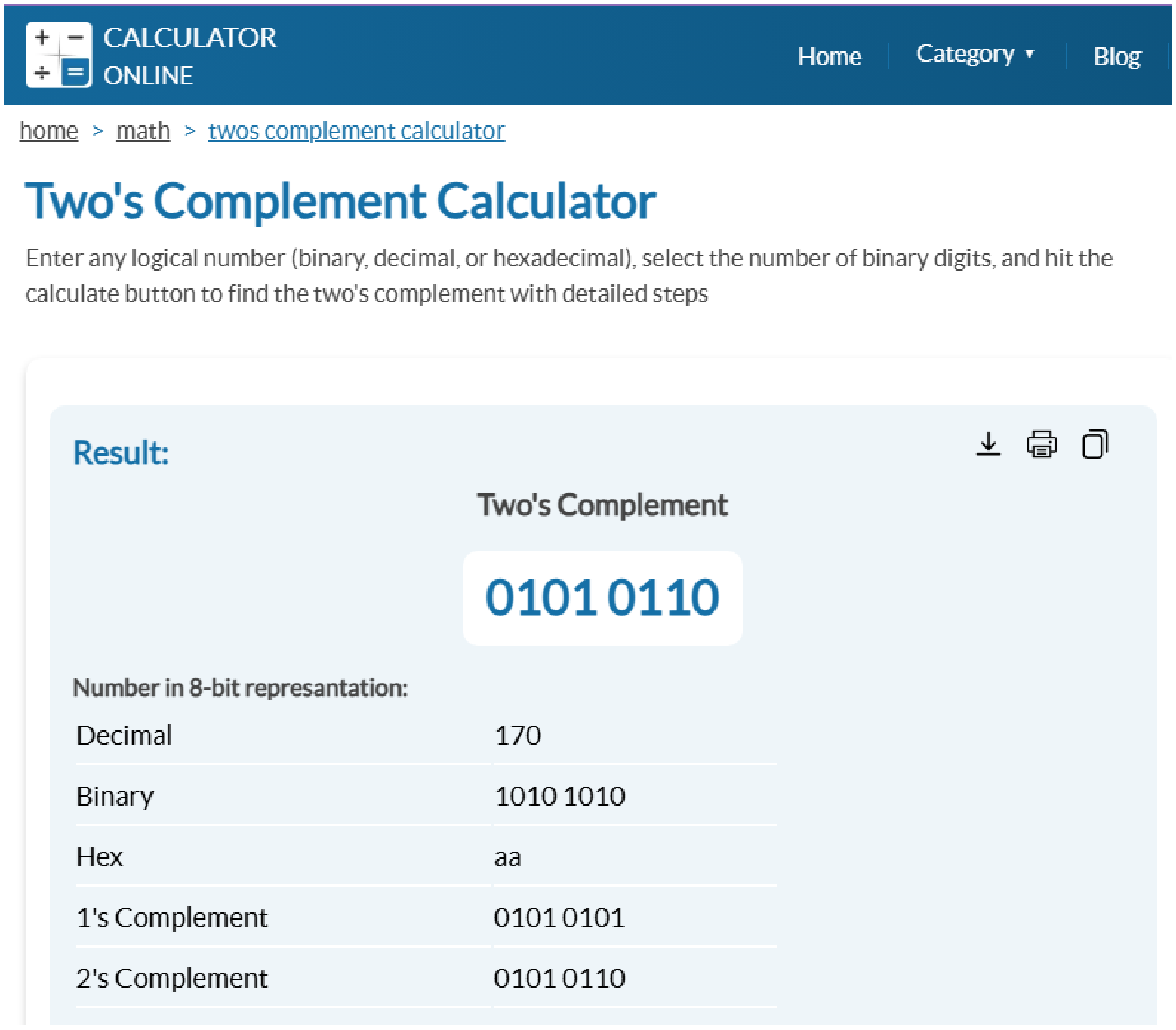
Result Copy PDF

Answer

2's Complement of Binary

0101 0110

Check one's complement



Two's Complement Calculator

Enter any logical number (binary, decimal, or hexadecimal), select the number of binary digits, and hit the calculate button to find the two's complement with detailed steps

Result:

Two's Complement

0101 0110

Number in 8-bit represantation:

Decimal	170
Binary	1010 1010
Hex	aa
1's Complement	0101 0101
2's Complement	0101 0110

Citations:

- “Two’s (2s) Complement Calculator.” 2025. Allmath.com. 2025. https://www.allmath.com/twos-complement.php.
- “Two’s (2s) Complement Calculator.” 2025. Calculator-Online.net. 2025. https://calculator-online.net/twos-complement-calculator/.

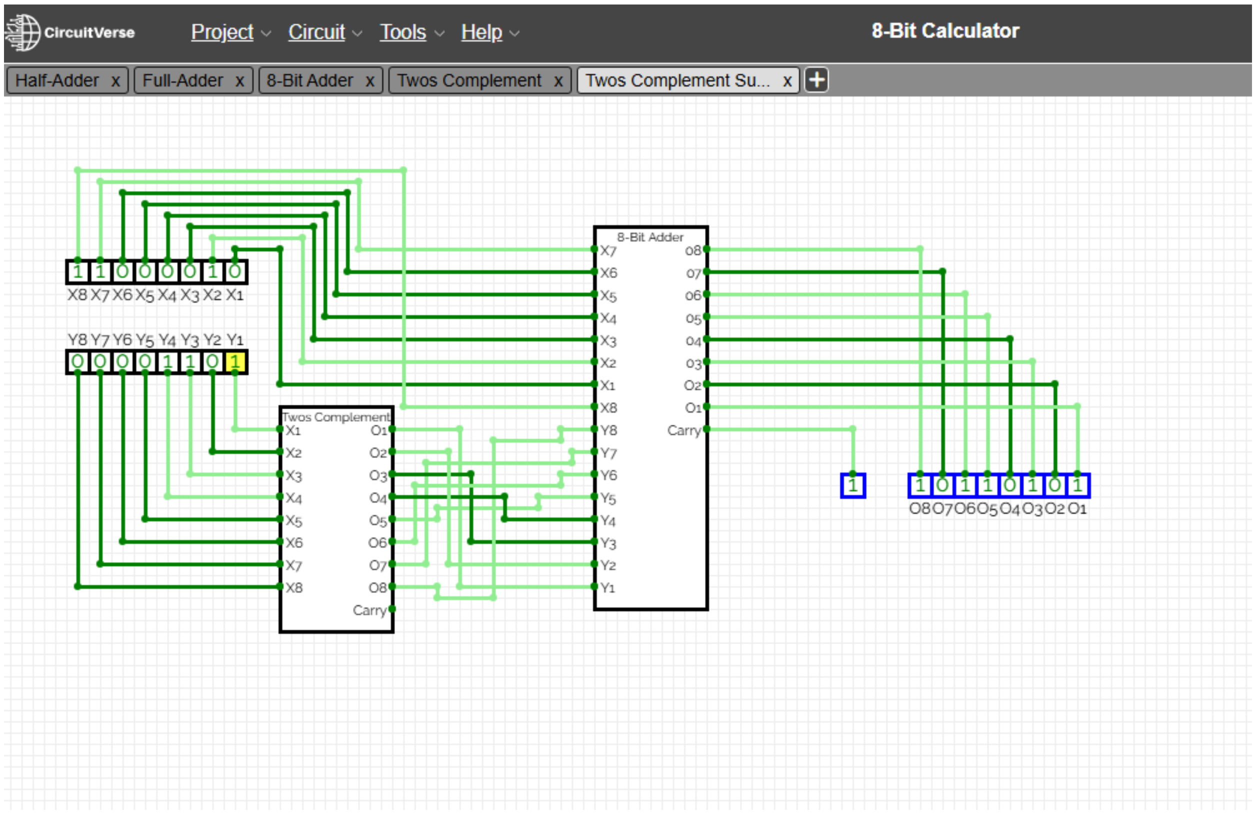
<u>Implement subtraction using the two's complement module alongside the adder:</u>

Using the Two's Complement module to make a subtraction module was done by feeding the minuend directly to the 8-bit adder. On the other hand, the subtrahend was led to the Two's complement first. Then it was fed to the 8-Bit Adder so that the minuend was added by the subtrahend which was changed to two's complement.

To test this module I used two separate online Two's Complement Calculators to verify my results. They both matched the output I received from my module and will be pasted below as well as cited.

Example with positive outcome:

11000010 (194) – 00001101 (13) = 10110101 (181)



Here A = 11000010, B = 00001101.
Find A - B = ? using 2's complement
First find 2's complement of B = 00001101

Note : 2's complement of a number is 1 added to it's 1's complement number.

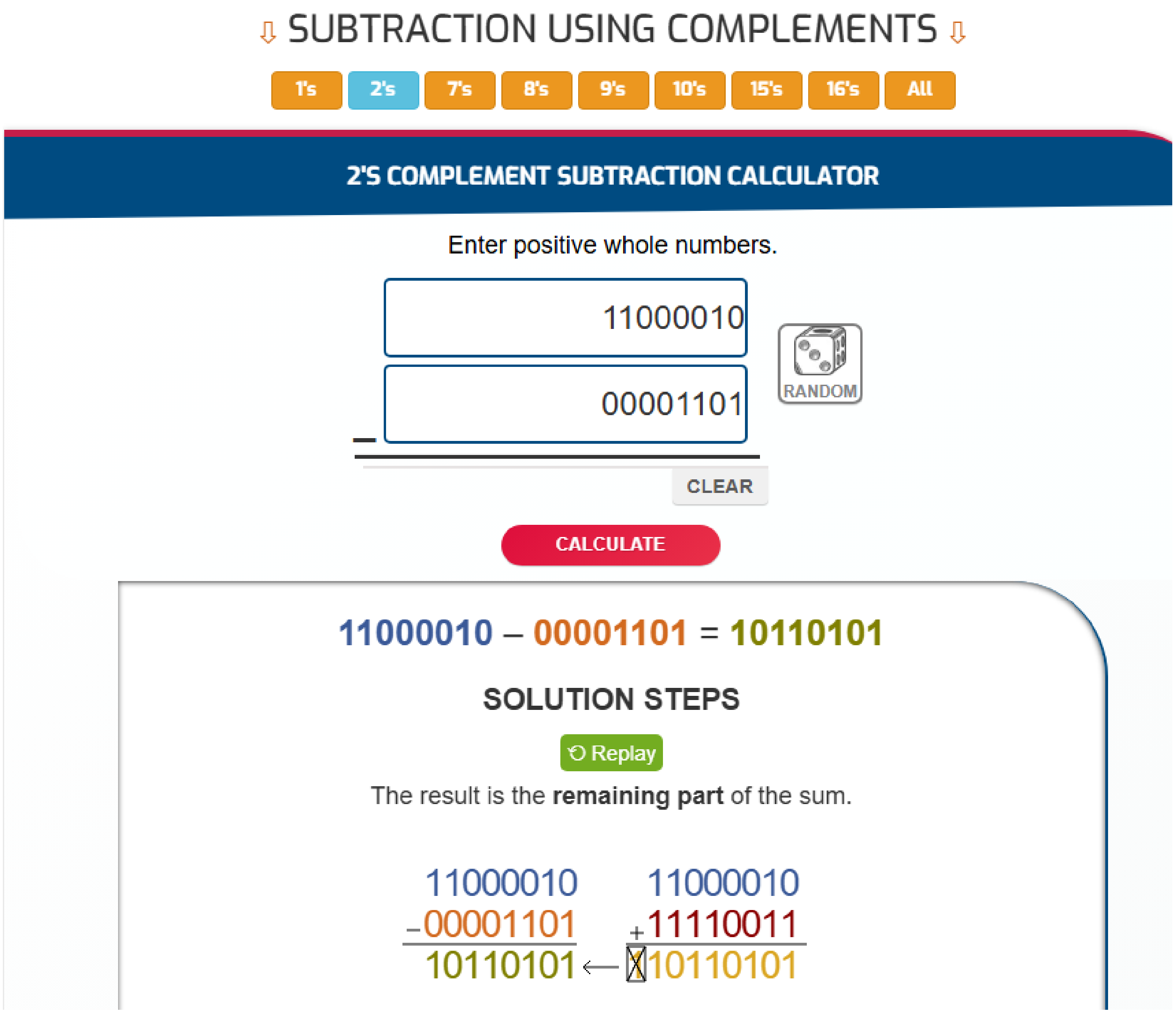
Step-1: 1's complement of 00001101 is obtained by subtracting each digit from 1

```
    1  1  1  1  1  1  1  1
 -  0  0  0  0  1  1  0  1
 --------------------------
    1  1  1  1  0  0  1  0
```

Step-2: Now add 1 to the 1's complement to obtain the 2's complement :
11110010 + 1 = 11110011

Step-3: Now Add this 2's complement of B to A

```
    1              1
    1  1  0  0  0  0  1  0
 +  1  1  1  1  0  0  1  1
 --------------------------
 1  1  0  1  1  0  1  0  1
```



⇩ SUBTRACTION USING COMPLEMENTS ⇩

1's 2's 7's 8's 9's 10's 15's 16's All

2'S COMPLEMENT SUBTRACTION CALCULATOR

Enter positive whole numbers.

```
  11000010
– 00001101
```

RANDOM CLEAR CALCULATE

11000010 – 00001101 = 10110101

SOLUTION STEPS

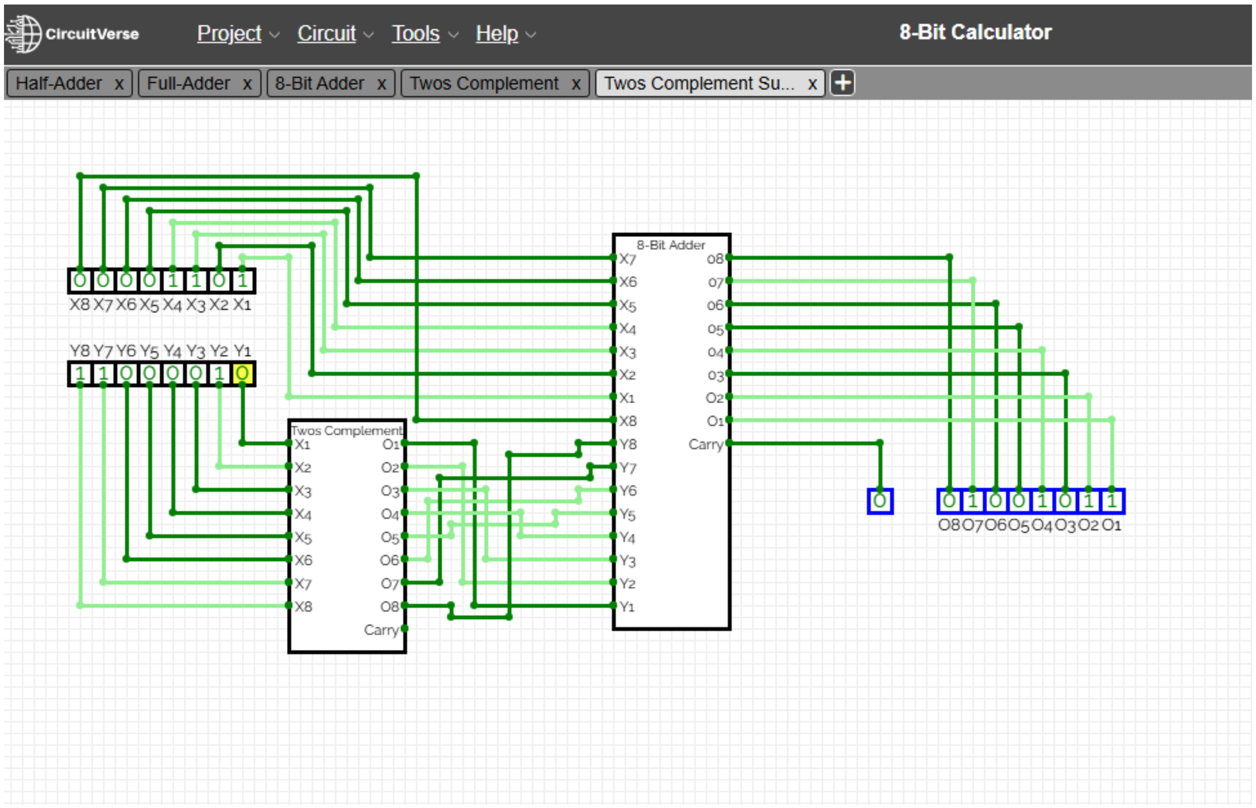
Replay

The result is the **remaining part** of the sum.

```
  11000010       11000010
 –00001101      +11110011
  10110101 ←── 110110101
```

Example with negative outcome:

00001101 (13) – 11000010 (194)


CircuitVerse
Project
Circuit
Tools
Help
8-Bit Calculator
Half-Adder x
Full-Adder x
8-Bit Adder x
Twos Complement x
Twos Complement Su... x
0 0 0 0 1 1 0 1
X8 X7 X6 X5 X4 X3 X2 X1
Y8 Y7 Y6 Y5 Y4 Y3 Y2 Y1
1 1 0 0 0 0 1 0
Twos Complement
8-Bit Adder
Carry
0
0 1 0 0 1 0 1 1
O8 O7 O6 O5 O4 O3 O2 O1

Solution:

2's complement subtraction steps :

Here A = 00001101, B = 11000010.
Find A - B = ? using 2's complement
First find 2's complement of B = 11000010

Note : 2's complement of a number is 1 added to it's 1's complement number.

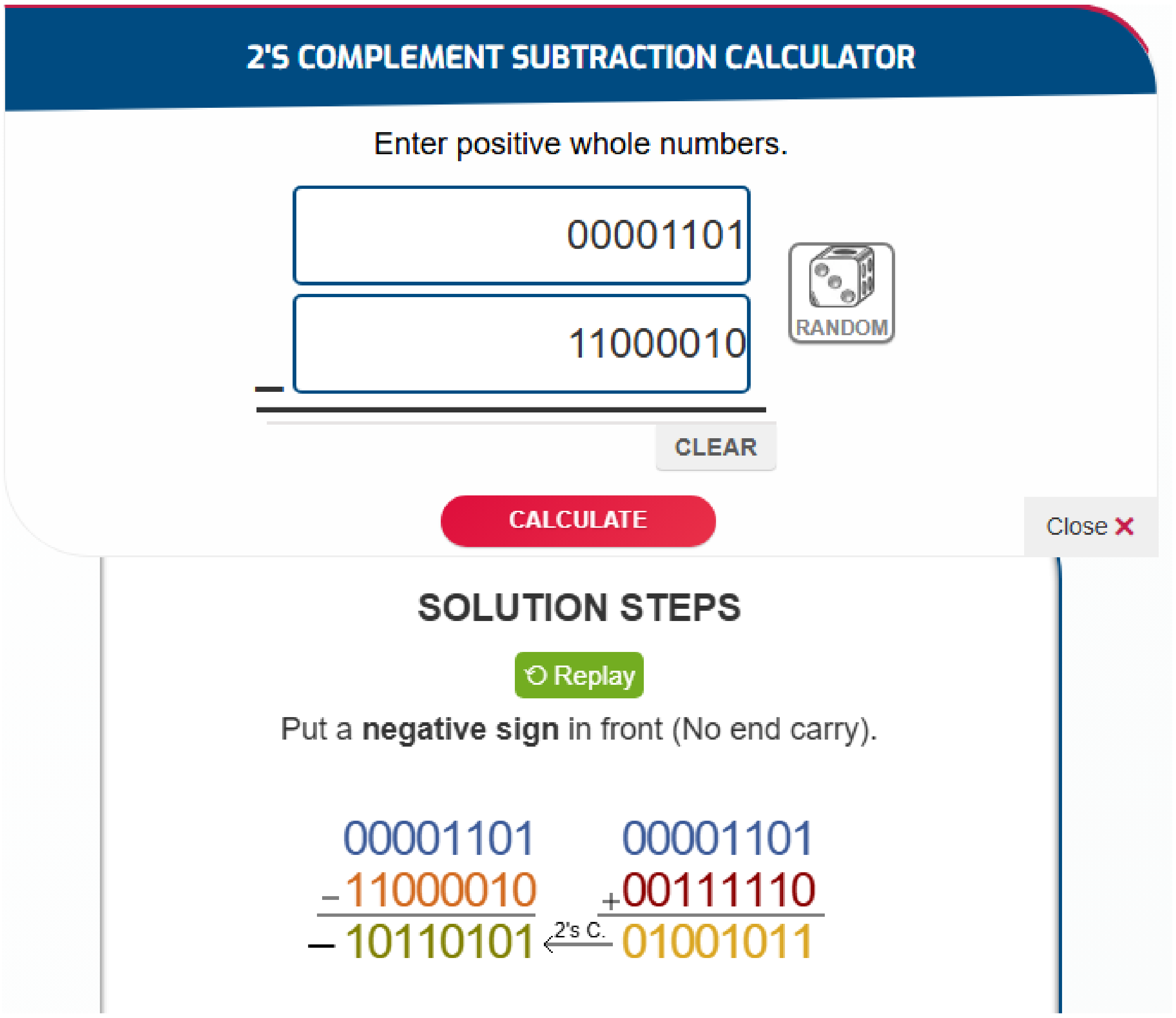
Step-1: 1's complement of 11000010 is obtained by subtracting each digit from 1

```
    1  1  1  1  1  1  1  1
 -  1  1  0  0  0  0  1  0
 -------------------------
    0  0  1  1  1  1  0  1
```

Step-2: Now add 1 to the 1's complement to obtain the 2's complement :
00111101 + 1 = 00111110

Step-3: Now Add this 2's complement of B to A

```
       1  1  1  1
    0  0  0  0  1  1  0  1
 +  0  0  1  1  1  1  1  0
 -------------------------
    0  1  0  0  1  0  1  1
```



Enter positive whole numbers.

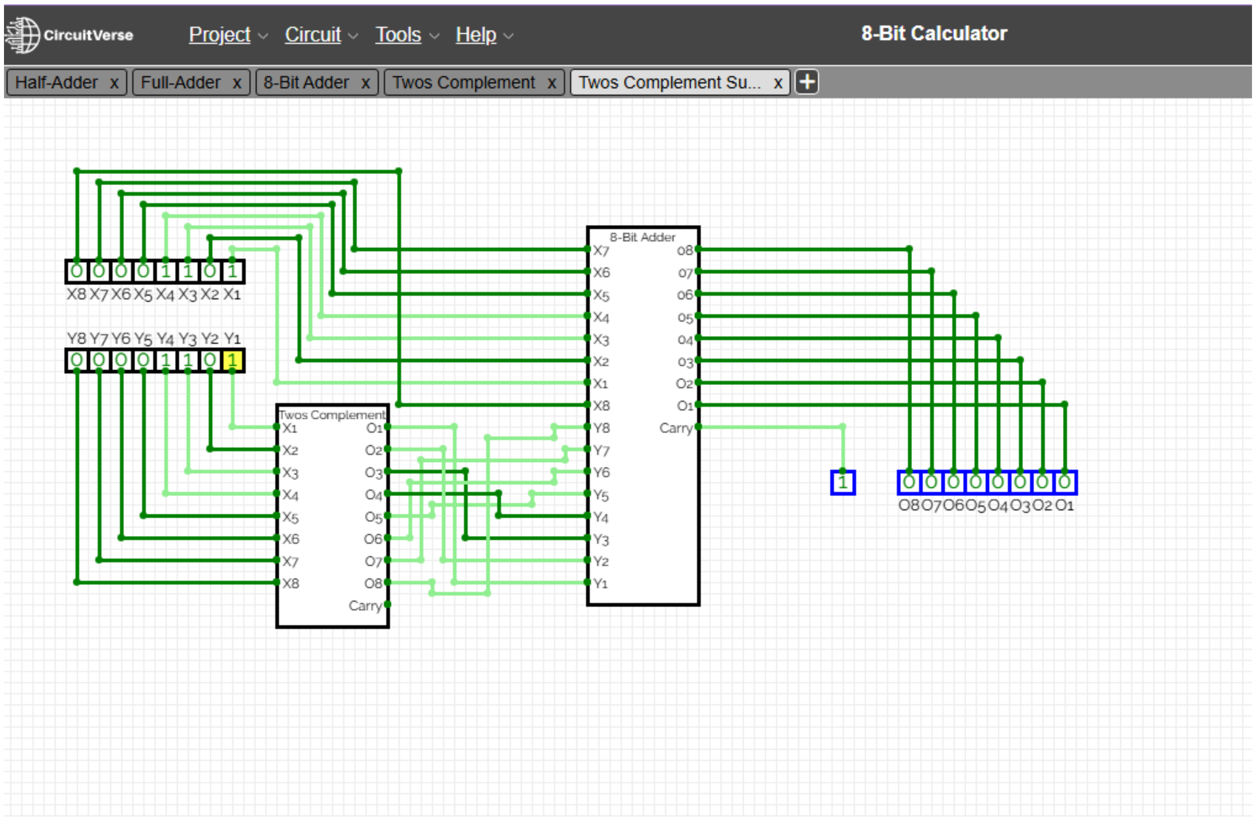
```
  00001101
− 11000010
```

SOLUTION STEPS

Put a **negative sign** in front (No end carry).

$$\begin{array}{r} 00001101 \\ -\,11000010 \\ \hline -\,10110101 \end{array} \xleftarrow{\text{2's C.}} \begin{array}{r} 00001101 \\ +\,00111110 \\ \hline 01001011 \end{array}$$

Example with the same number subtracting itself:



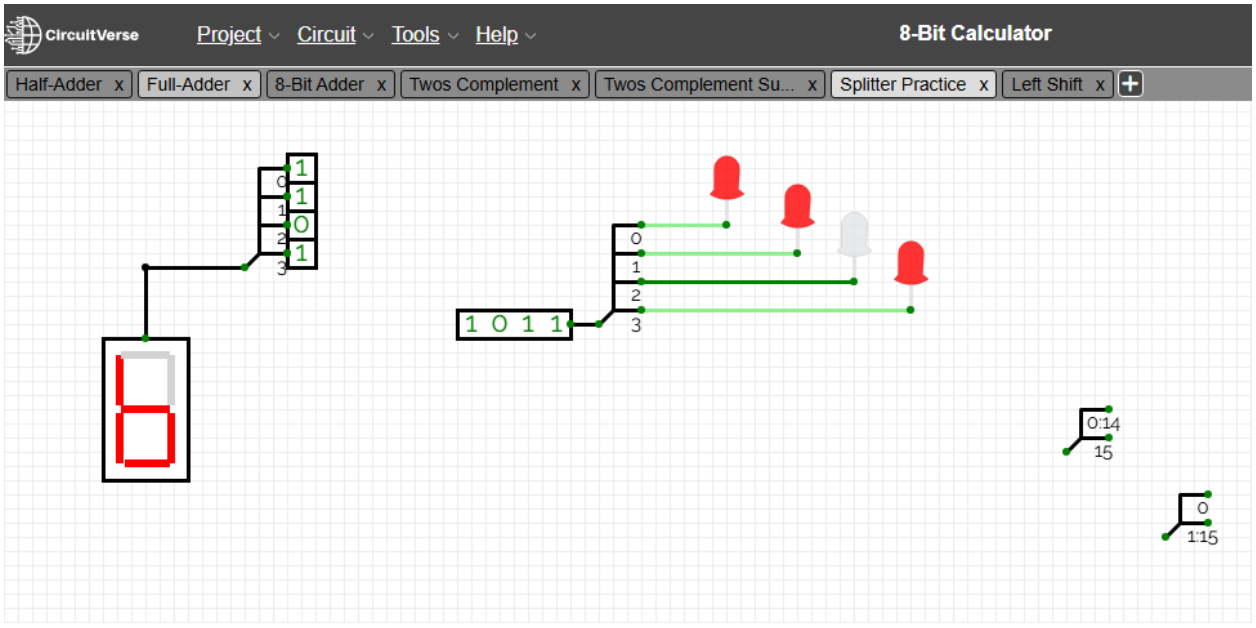
Citations:

- Shah, Piyush N. 2024. “2’S Complement Subtraction Calculator.” Atozmath.com. 2024. https://atozmath.com/NumberSubComp.aspx?q=2&m=1&q1=1%6000001101%6011000010%602&do=1#tblSolution.
- Math, Mad for. 2025. “2’S COMPLEMENT SUBTRACTION CALCULATOR.” Madformath.com. 2025. https://madformath.com/calculators/digital-systems/complement-subtraction/2-s-complement-subtraction-calculator/2-s-complement-subtraction-calculator.

Design a multiplication module

A shift-and-add algorithm is needed to examine each bit of the multiplier to create the multiplication module. Bit shifting to the left is an advantageous operation for multiplication as it takes it to the position of the next higher power of two. This has the effect of multiplying the integer by 2.

To shift a binary number to left, a module will needed to be made. At first, I concluded that this would be another messy circuit with wires all over the place. I knew there had to be some easier way to complete this. So I went over more tutorials for circuits and discovered the splitter. It can be used when there are too many wires laying in parallel to each other. The splitter can combine the wires into one bundle. Essentially it can divide a signal into multiple identical copies into a single output. For the previous modules this would have been extremely helpful as I also found out that the input bandwidth in CircuitVerse can be increased. Instead of individually adding and connecting bits to their respective places, I could have increased the bandwidth of a singular input and used a splitter to the signal from the input to multiple outputs. Practice done in Circuit verse can be found below.

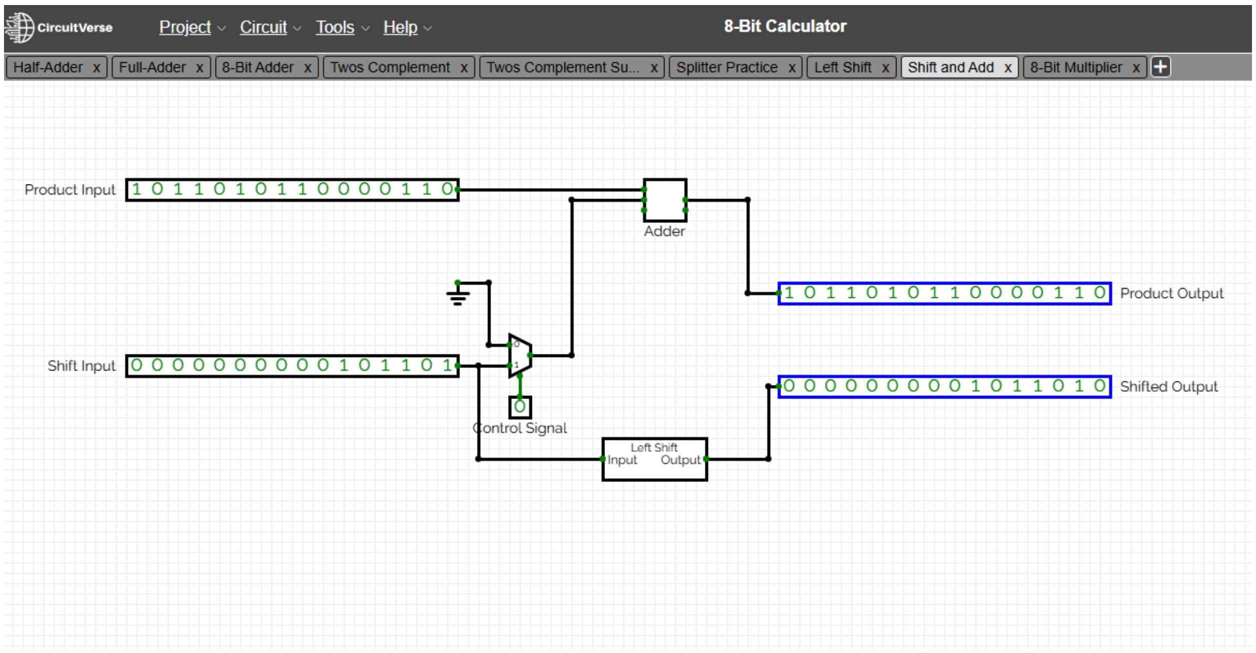


This discovery can help with the shifter because not only does it remove the need to add multiple inputs and wire them to their location, but it also helps me select which inputs go through and which become the output.

To start the left shift module, an input of 16-bit bandwidth is taken in and led through a splitter which separates the leftmost bit from the rest of the 15 other bits. Then it is led through another splitter where it takes in a ground input which represents 0. This is done because whenever a bit is shifted to the left, the leftmost bit is removed, then the following bits are shifted left, and finally the rightmost bit is represented by 0 making a successful shift.

This is not all that is needed to make the 8-bit multiplier. To successfully make the 8-bit multiplier, a module to process bit multiplication is needed first. This is done by the shift-and-add algorithm. To begin, the product input is led to an in-application adder which is handy for handling 16 bit inputs. This is the first input for the adder. The second input is the shift input, but it needs to undergo a control. Here, a multiplexer is introduced, it's purpose is to take in multiple inputs and only allow one output based on the control signal state. Depending on whether it is 1 or 0, it will allow the input with the corresponding control signal value to pass through, where it is then led to the adder with the product input. This means that each bit that is 1, the multiplicand is not only shifted but is also added to the accumulator. After the adder works with the two inputs it receivers, its output results in the product output. Now we go back to the shift input, this will pass through the left shift subcircuit. Thus, the two outputs for the bit multiplication module are the product output and the shift output.

A picture of the circuit with shifting and addition stage producing expected binary results will be pasted below:

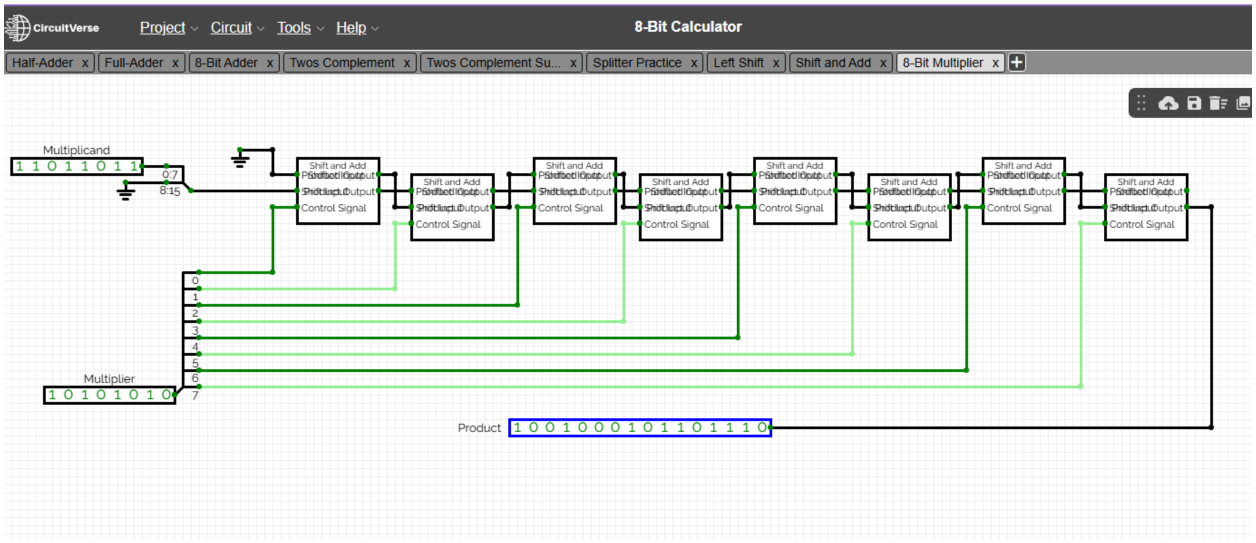


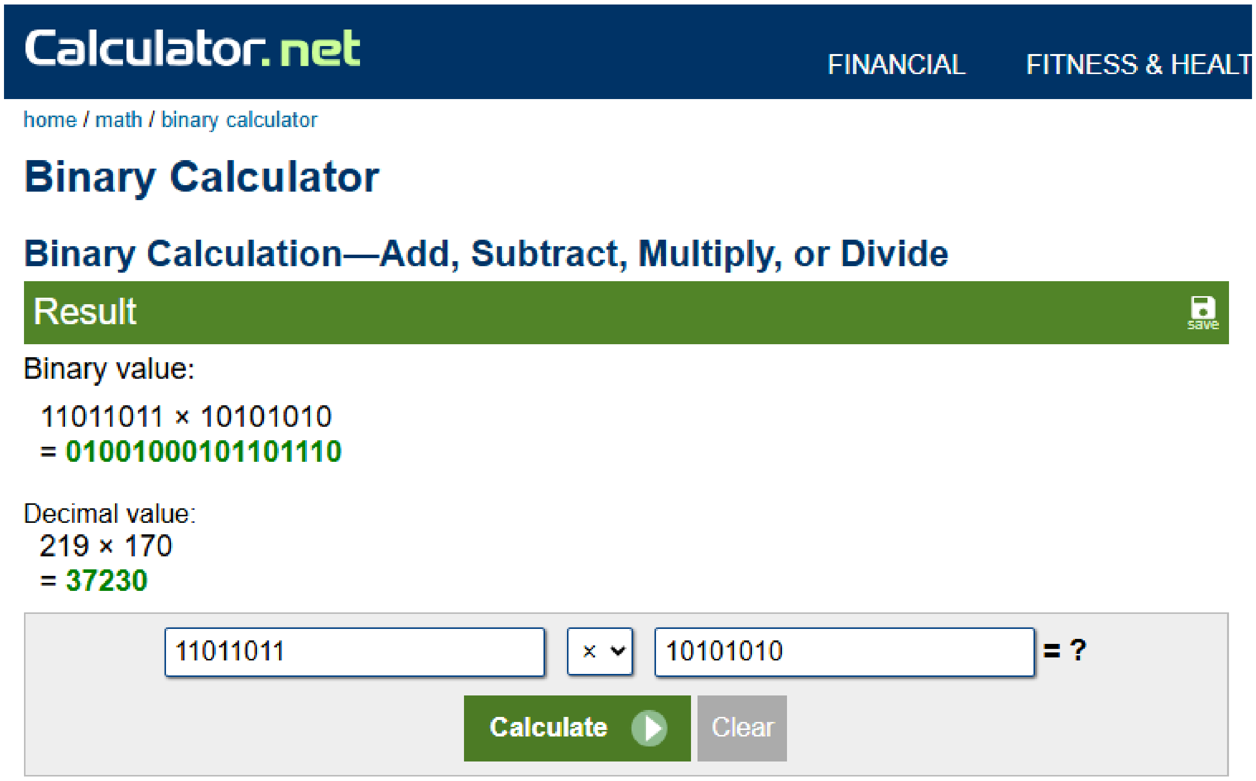
Finally, comes the construction of the 8-bit multiplier. Here the shift and add module is used eight times to examine each bit and depending on the control signal received will accumulate the shift input with the product input or only pass the product input. This process is repeatedly done until the final product is shown as the final output.

Multiplication with larger numbers:

Multiplication with small numbers:

Example:





home / math / binary calculator

Binary Calculator

Binary Calculation—Add, Subtract, Multiply, or Divide

Result

Binary value:

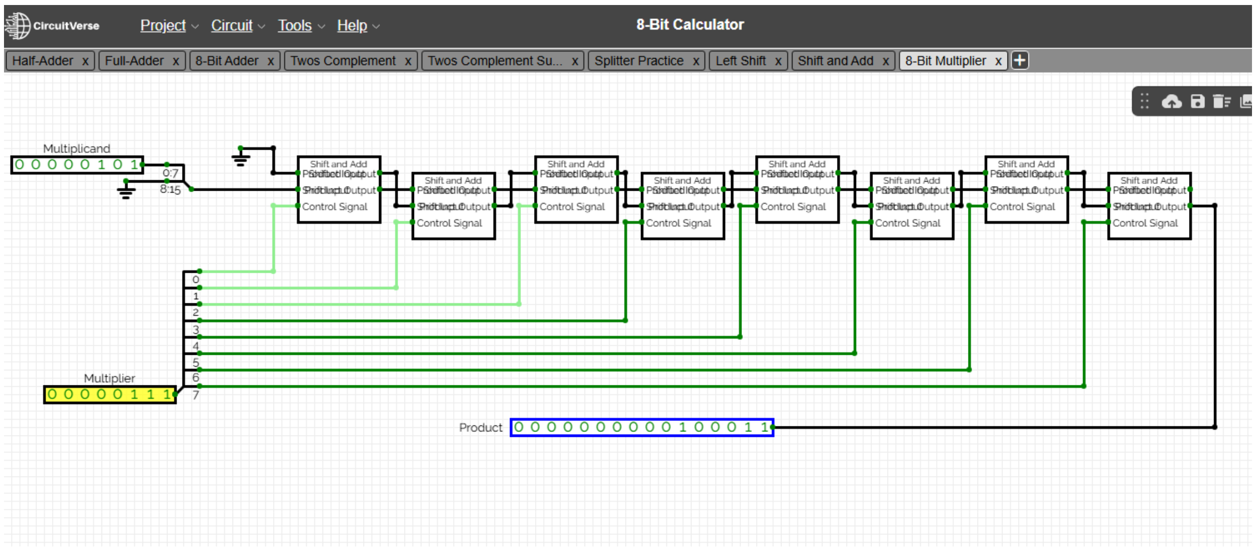
11011011 × 10101010
= 01001000101101110

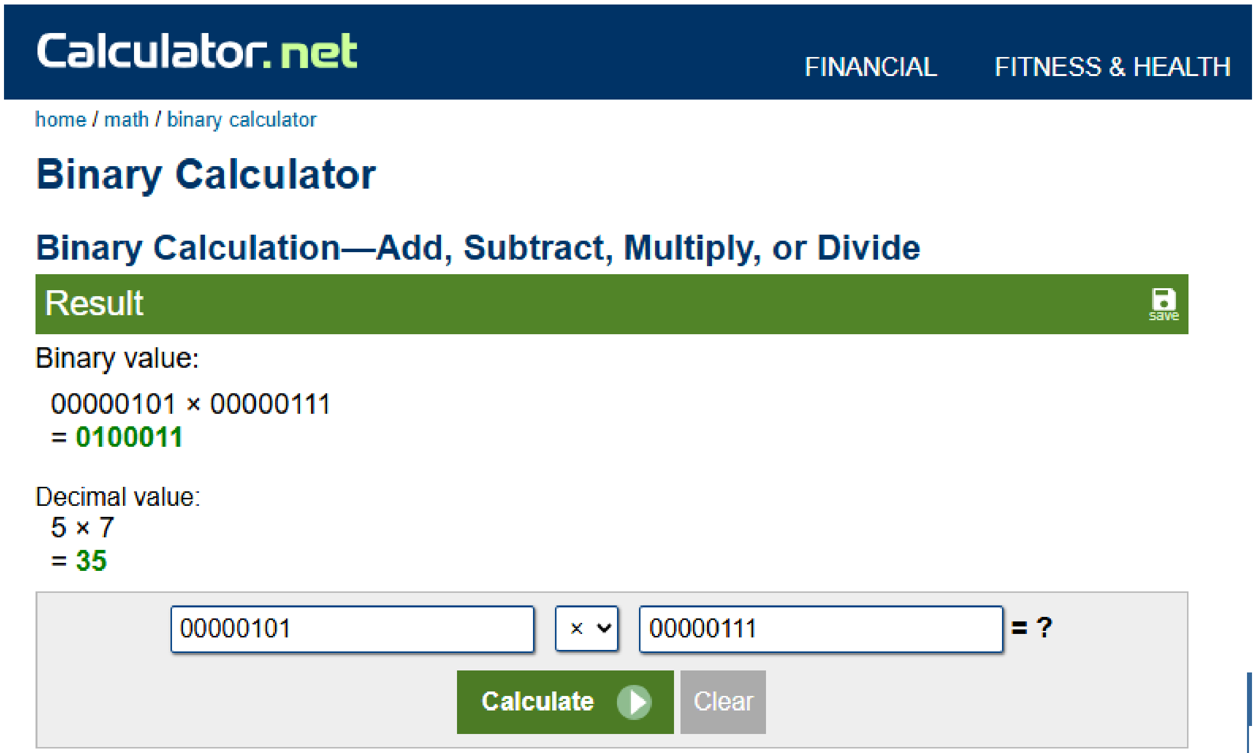
Decimal value:
219 × 170
= 37230

11011011 × 10101010 = ?

Calculate Clear

Example 1:





home / math / binary calculator

Binary Calculator

Binary Calculation—Add, Subtract, Multiply, or Divide

Result

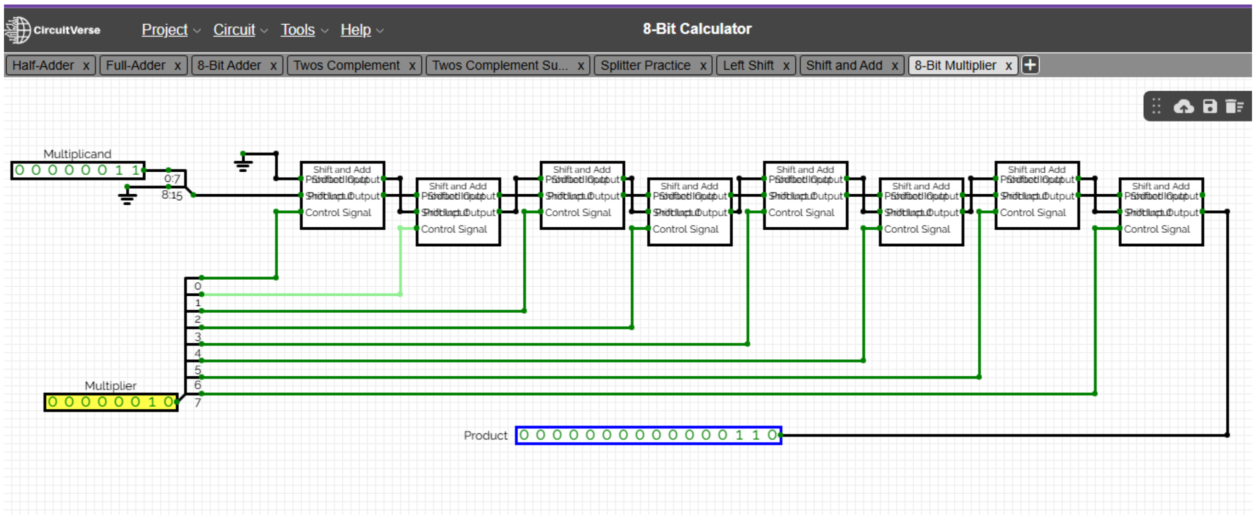
Binary value:

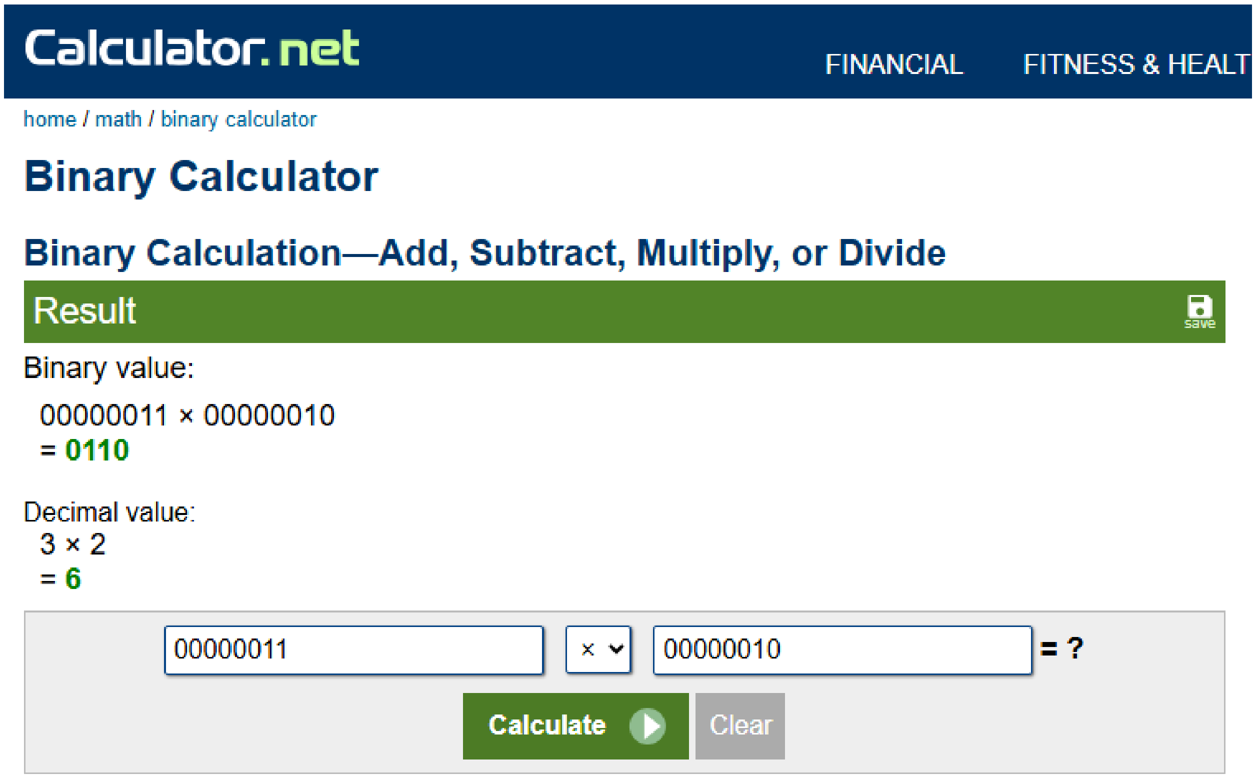
00000101 × 00000111
= 0100011

Decimal value:
5 × 7
= 35

00000101 × 00000111 = ?

Example 2:





Calculator.net

FINANCIAL FITNESS & HEALT

home / math / binary calculator

Binary Calculator

Binary Calculation—Add, Subtract, Multiply, or Divide

Result

Binary value:

00000011 × 00000010
= 0110

Decimal value:
3 × 2
= 6

00000011 × 00000010 = ?

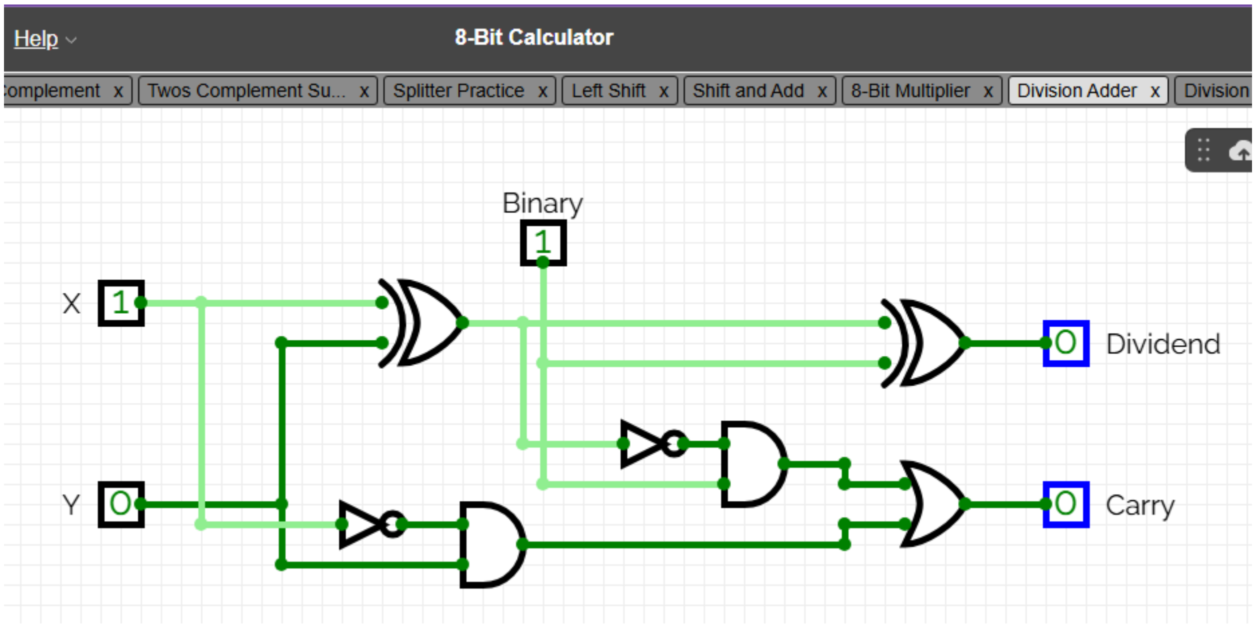
Calculate Clear

Citation:

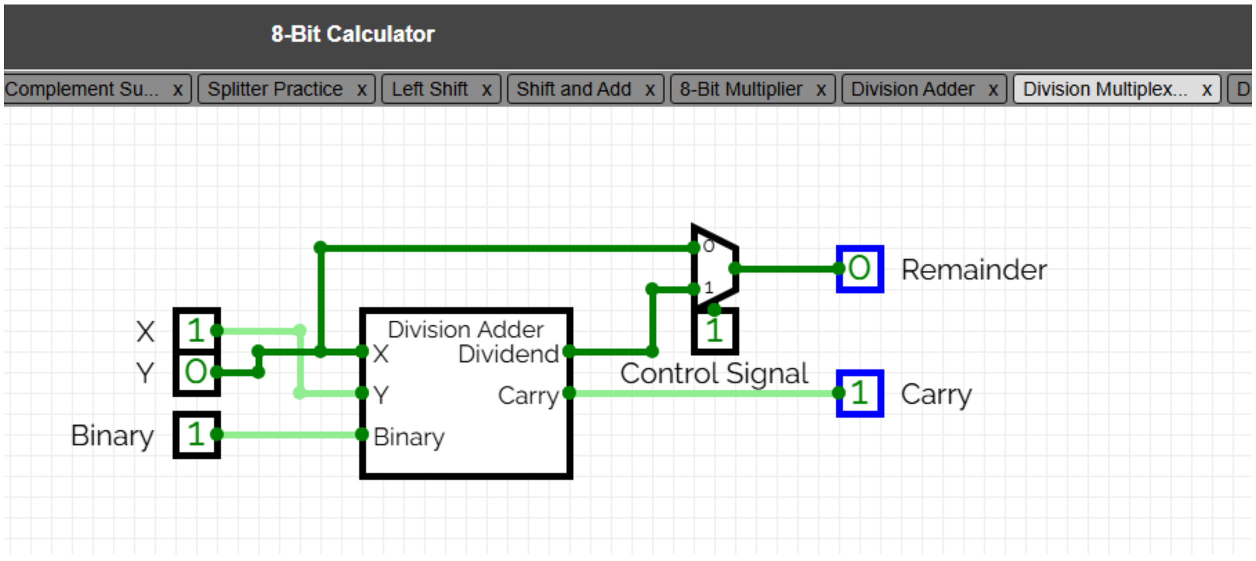
“Binary Calculator.” 2025. Calculator.net. 2025. https://www.calculator.net/binary-calculator.html.

Create a Division Circuit

To begin with, a repeated subtraction method is needed to implement a division circuit on the calculator. First, a special modified full-adder is made for the division module. The full adder I already made has been modified to include NOT gates, which are needed to invert the value of the two first inputs for the two AND gates. This leaves us with dividends and carry bits.



With these two bits, we will create a custom multiplexer for division where we take in an X, Y, Binary, and Control Signal input. The X,Y and Binary inputs are run through the custom division adder made before. It's output will be used as a carry output and will also be run through an in-program multiplexer. This multiplexer in the circuit is used to filter the output of either dividend from division adder or Y based on the input Control Signal. When the control signal is 1, the dividend will be allowed to pass through. When it is 0, the Y will be passed through and depending on which is passed, will be the remainder.



To perform 8-bit binary division, the Division Multiplexer module is iterated 16 times to compute both the quotient and remainder. The module accepts an 8-bit dividend, which is distributed into two parts, the four most significant bits which are routed to the upper row of four division multiplexers, and the three least significant bits which are routed to the rightmost three multiplexers. The eighth bit of the dividend is directed into an OR gate, whose output serves a critical control function explained later.

The 8-bit divisor is fed into the top input (X) of the multiplexers. Each bit of the divisor is connected to four multiplexers that align with its column, ensuring accurate bit-level comparison and manipulation across all stages.

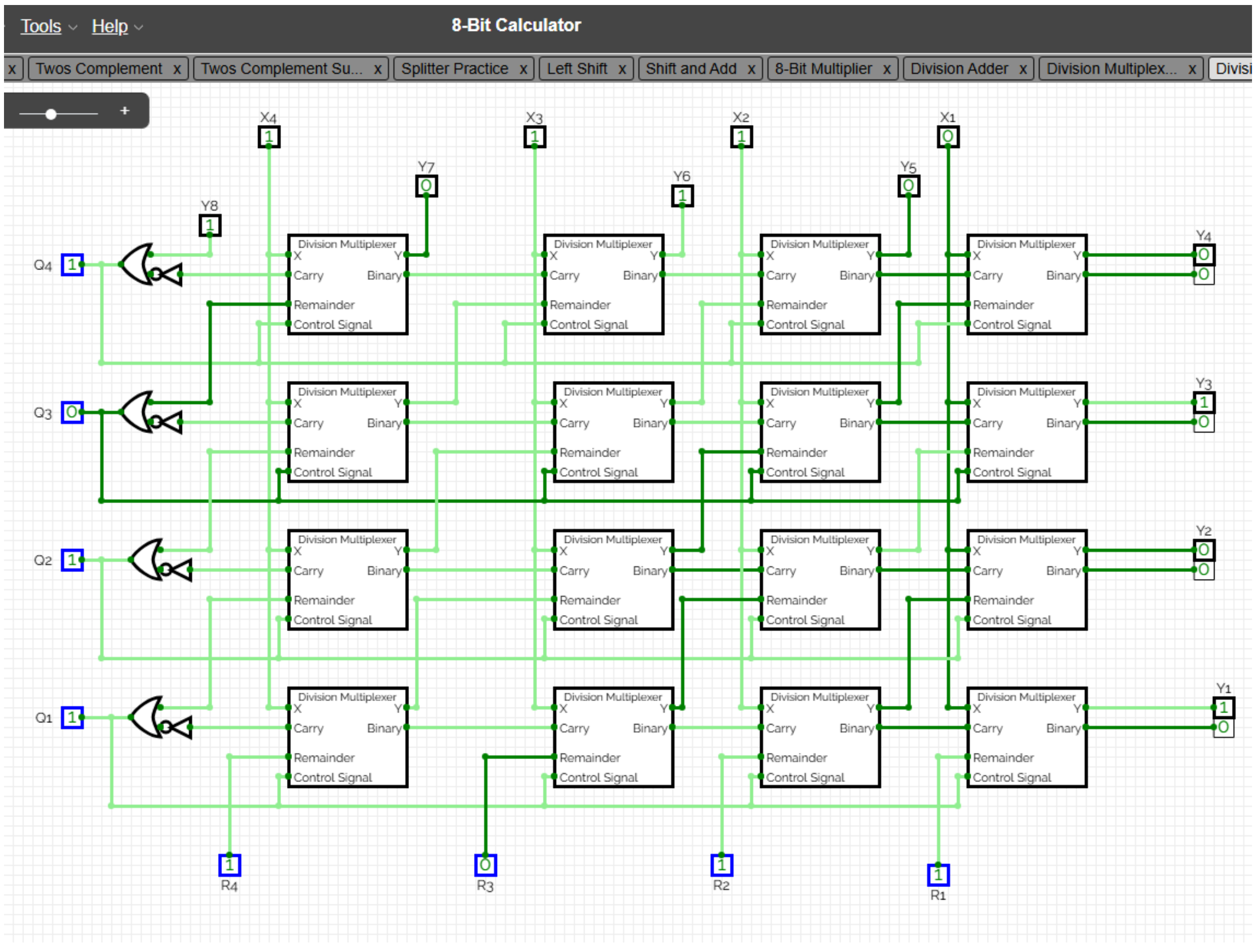
A constant binary input of four zeros is connected to the X inputs of the four rightmost multiplexers. These constants support initialization and boundary conditions during the iterative division process.

The control signal for each multiplexer originates from the aforementioned OR gate. This OR gate receives two inputs, the eighth bit of the dividend and the inverted carry-out signal from the leftmost multiplexer in the current row. Once evaluated, the result of the OR gate is propagated across all control inputs for the multiplexers on that row.

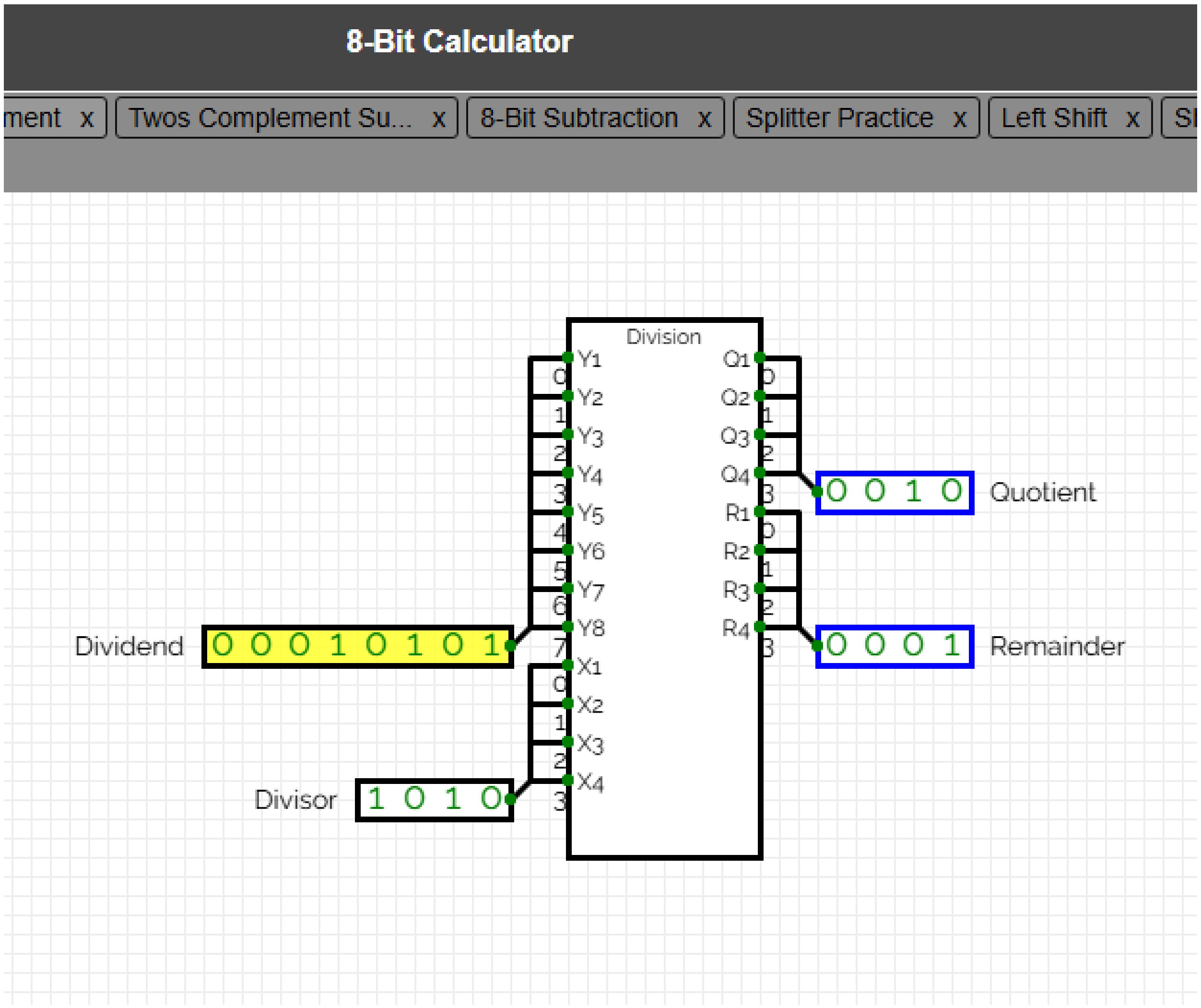
The carry output from each multiplexer is fed into the binary input of the adjacent multiplexer to its left, forming a cascading structure. This continues until the signal reaches the OR gate, completing one cycle of quotient evaluation.

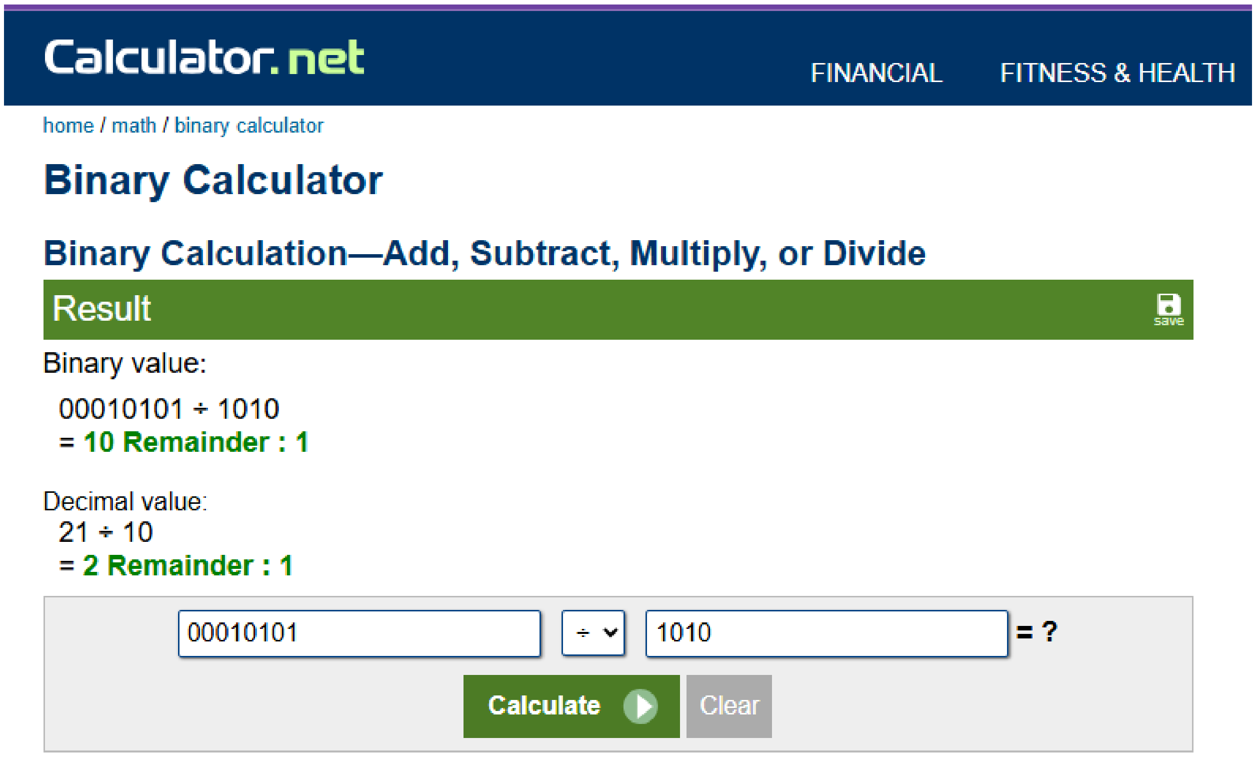
This cascading process occurs across four horizontal levels, each producing a single bit of the quotient. Collectively, this yields the 4-bit quotient.

Regarding the remainder, each partial remainder is passed to the Y input of the multiplexer diagonally below and to the left. This operation continues across the circuit until there are no further multiplexers available, resulting in a final 4-bit remainder.



Here is the 8-bit Divider in action with an example backed by an online source





Binary Calculator

Binary Calculation—Add, Subtract, Multiply, or Divide

Result

Binary value:

00010101 ÷ 1010
= 10 Remainder : 1

Decimal value:
21 ÷ 10
= 2 Remainder : 1

00010101 ÷ 1010 = ?

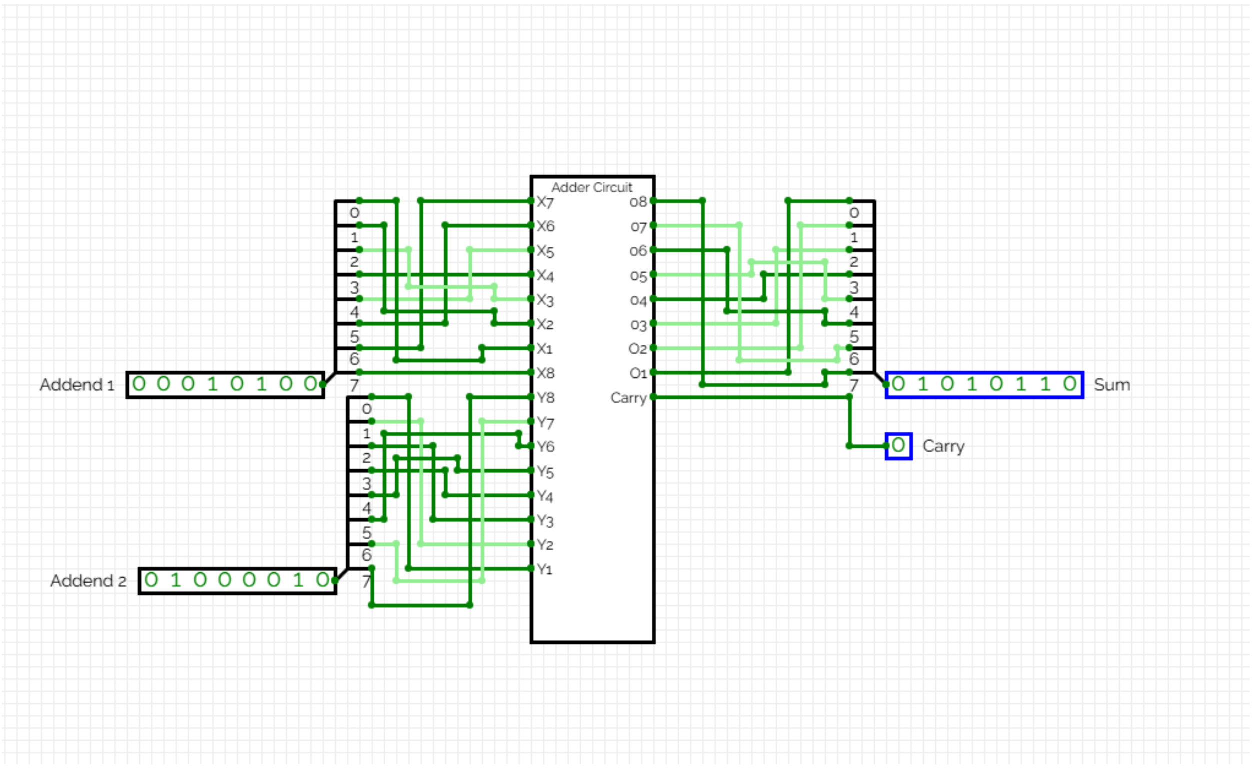
Citations:

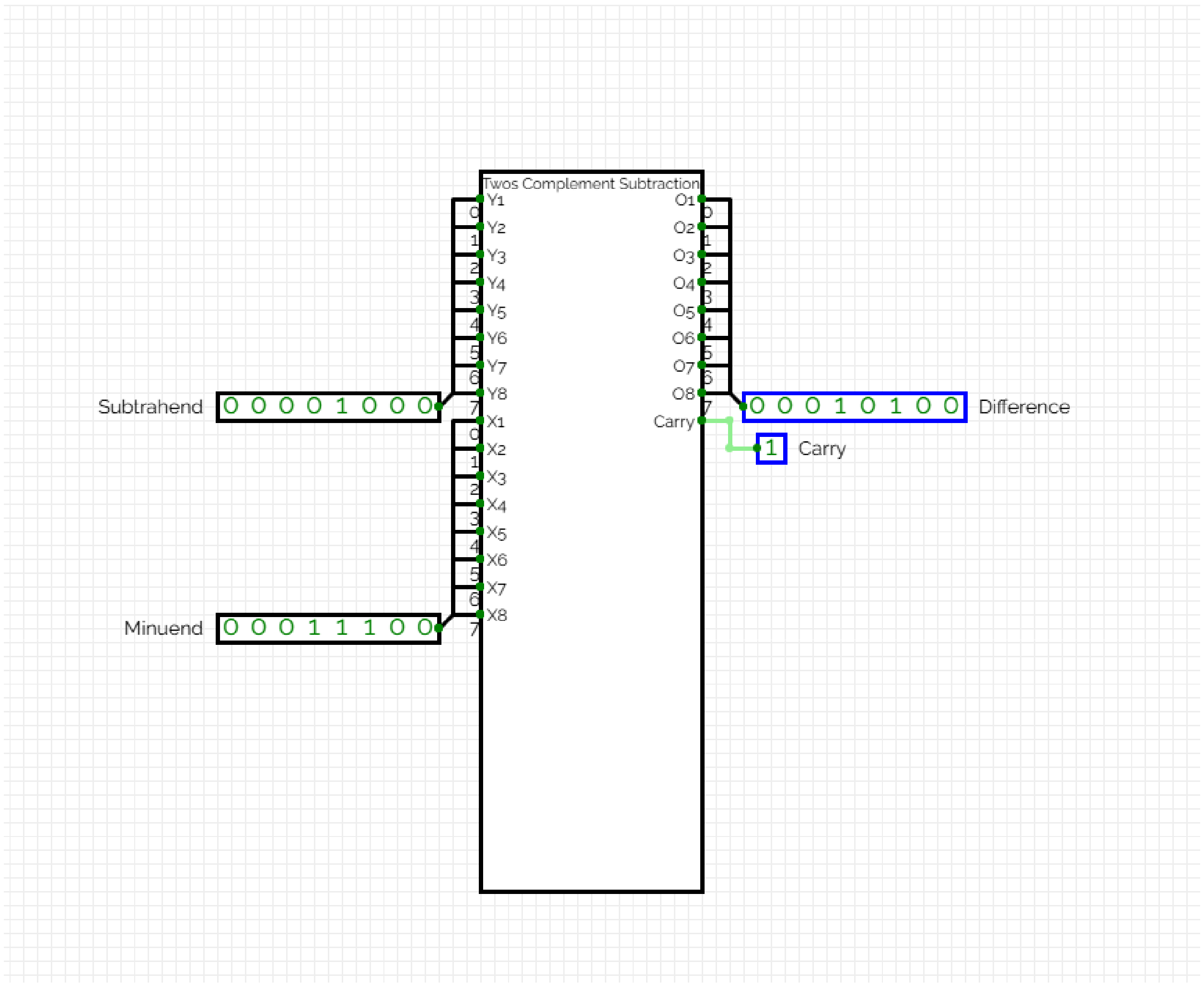
"Binary Calculator." 2025. Calculator.net. 2025. https://www.calculator.net/binary-calculator.html?number1=00010101&c2op=%2F&number2=1010&calctype=op&x=Calculate.

Integrate the Modules into a complete calculator

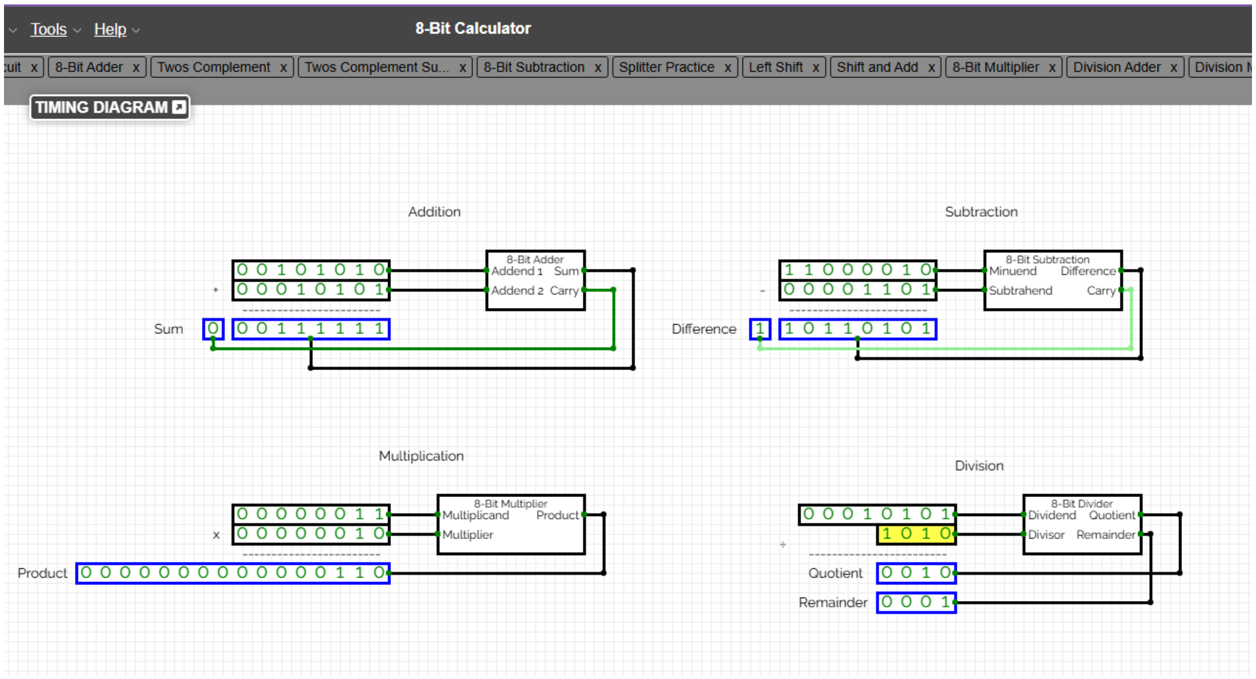
Before beginning this portion of the project, I realized that the previously constructed circuits for addition and subtraction appeared disorganized when integrated as subcircuits within the larger calculator module. In an attempt to resolve this, I modified the layout of the full adder subcircuit. However, this change inadvertently affected all other circuits where the subcircuit had been utilized, and any modification to a subcircuit propagates to all instances where it is used.

To avoid unintended disruptions to other components, I decided to revert the changes and instead create new, dedicated circuits for the 8-bit Adder and 8-bit Subtraction modules. This approach ensures clearer implementation within the calculator module and prevents interference with other existing circuits. The newly created modules are presented below.





With these alterations made, making the calculator module was a sinch. It will be pasted below and include examples previously used in the past sections of the project document.


Tools
Help
8-Bit Calculator
uit x
8-Bit Adder x
Twos Complement x
Twos Complement Su... x
8-Bit Subtraction x
Splitter Practice x
Left Shift x
Shift and Add x
8-Bit Multiplier x
Division Adder x
Division
TIMING DIAGRAM
Addition
0 0 1 0 1 0 1 0
+
0 0 0 1 0 1 0 1
8-Bit Adder
Addend 1
Sum
Addend 2
Carry
Sum
0
0 0 1 1 1 1 1 1
Subtraction
1 1 0 0 0 0 1 0
-
0 0 0 0 1 1 0 1
8-Bit Subtraction
Minuend
Difference
Subtrahend
Carry
Difference
1
1 0 1 1 0 1 0 1
Multiplication
0 0 0 0 0 0 1 1
x
0 0 0 0 0 0 1 0
8-Bit Multiplier
Multiplicand
Product
Multiplier
Product
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
Division
0 0 0 1 0 1 0 1
÷
1 0 1 0
8-Bit Divider
Dividend
Quotient
Divisor
Remainder
Quotient
0 0 1 0
Remainder
0 0 0 1