Prime Number Identifier Circuit Testing Using CircuitVerse with Maxima Code included

Author: Arsalan Khan 210862640
Date: 2023-10-27

Introduction

The objective of this lab exercise was to design and simulate a digital circuit capable of identifying prime numbers within a set range. A prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. The identification of prime numbers is a critical function in various fields, including cryptography and systems security. In this report, the circuit's logic is derived from a Boolean expression formulated to detect prime numbers within a 4-bit binary input, representing the decimal numbers 0 to 15.

Logic Equation:

$$prime = \overline{a_3}\ \overline{a_2}\ a_1 + \overline{a_3}\ a_2\ a_0 + \overline{a_2}\ a_1\ a_0 + a_2\ \overline{a_1}\ a_0$$

A3 is the MSB and a0 is the LSB and the expected resulted to come true using this equation is 2,3,5,7,11,13.

Truth Table:

a3	a2	a1	a0	Binary	Decimal	Prime?
0	0	0	0	0	0	No
0	0	0	1	1	1	No
0	0	1	0	10	2	Yes
0	0	1	1	11	3	Yes
0	1	0	0	100	4	No
0	1	0	1	101	5	Yes
0	1	1	0	110	6	No
0	1	1	1	111	7	Yes
1	0	0	0	1000	8	No
1	0	0	1	1001	9	No
1	0	1	0	1010	10	No
1	0	1	1	1011	11	Yes
1	1	0	0	1100	12	No
1	1	0	1	1101	13	Yes
1	1	1	0	1110	14	No
1	1	1	1	1111	15	No

Maxima Code from Previous Lab:

This document outlines the testing procedure of the Prime Number Identifier Circuit using Maxima. The primary goal is to verify the correctness of the logic equation by running it through all possible binary input combinations from 0000 to 1111.

Equation Definitions:

```
t1: (not a3) and (not a2) and a1;
t2: (not a3) and a2 and a0;
t3: (not a2) and a1 and a0;
t4: a2 and (not a1) and a0;

prime: t1 or t2 or t3 or t4;
```

Testing Procedure:

The logic equation was inputted into the Maxima online calculator. Each binary combination was tested, and the outputs were documented.

Maxima Session Code (input):

```
t1: (not a3) and (not a2) and a1;
t2: (not a3) and a2 and a0;
t3: (not a2) and a1 and a0;
t4: a2 and (not a1) and a0;
prime: t1 or t2 or t3 or t4;

/* Testing all 16 combinations */
prime, a0=false, a1=false, a2=false, a3=false;
prime, a0=true, a1=false, a2=false, a3=false;
prime, a0=false, a1=true, a2=false, a3=false;
prime, a0=true, a1=true, a2=false, a3=false;
prime, a0=false, a1=false, a2=true, a3=false;
prime, a0=true, a1=false, a2=true, a3=false;
prime, a0=false, a1=true, a2=true, a3=false;
prime, a0=true, a1=true, a2=true, a3=false;
prime, a0=false, a1=false, a2=false, a3=true;
prime, a0=true, a1=false, a2=false, a3=true;
prime, a0=false, a1=true, a2=false, a3=true;
prime, a0=true, a1=true, a2=false, a3=true;
prime, a0=false, a1=false, a2=true, a3=true;
prime, a0=true, a1=false, a2=true, a3=true;
prime, a0=false, a1=true, a2=true, a3=true;
prime, a0=true, a1=true, a2=true, a3=true;
```

Transcript(Output):

```
(%i1)

t1: (not a3) and (not a2) and a1;
(%o1) ¬a3∧¬a2∧a1

(%i2)

t2: (not a3) and a2 and a0;
(%o2) ¬a3∧a2∧a0

(%i3)

t3: (not a2) and a1 and a0;
(%o3) ¬a2∧a1∧a0

(%i4)

t4: a2 and (not a1) and a0;
(%o4) a2∧¬a1∧a0

(%i5)

prime: t1 or t2 or t3 or t4;
(%o5) ¬a3∧¬a2∧a1∨¬a3∧a2∧a0∨¬a2∧a1∧a0∨a2∧¬a1∧a0

(%i6)

/* Testing all 16 combinations */
prime, a0=false, a1=false, a2=false, a3=false;
(%o6) false

(%i7)

prime, a0=true, a1=false, a2=false, a3=false;
(%o7) false

(%i8)

prime, a0=false, a1=true, a2=false, a3=false;
(%o8) true
```

```
(%i9)

prime, a0=true, a1=true, a2=false, a3=false;
(%o9) true

(%i10)

prime, a0=false, a1=false, a2=true, a3=false;
(%o10)      false

(%i11)

prime, a0=true, a1=false, a2=true, a3=false;
(%o11)      true

(%i12)

prime, a0=false, a1=true, a2=true, a3=false;
(%o12)      false

(%i13)

prime, a0=true, a1=true, a2=true, a3=false;
(%o13)      true

(%i14)

prime, a0=false, a1=false, a2=false, a3=true;
(%o14)      false

(%i15)

prime, a0=true, a1=false, a2=false, a3=true;
(%o15)      false

(%i16)

prime, a0=false, a1=true, a2=false, a3=true;
(%o16)      false
```

```
(%i17)

prime, a0=true, a1=true, a2=false, a3=true;
(%o17)      true

(%i18)

prime, a0=false, a1=false, a2=true, a3=true;
(%o18)      false

(%i19)

prime, a0=true, a1=false, a2=true, a3=true;
(%o19)      true

(%i20)

prime, a0=false, a1=true, a2=true, a3=true;
(%o20)      false

(%i21)

prime, a0=true, a1=true, a2=true, a3=true;
(%o21)      false
```

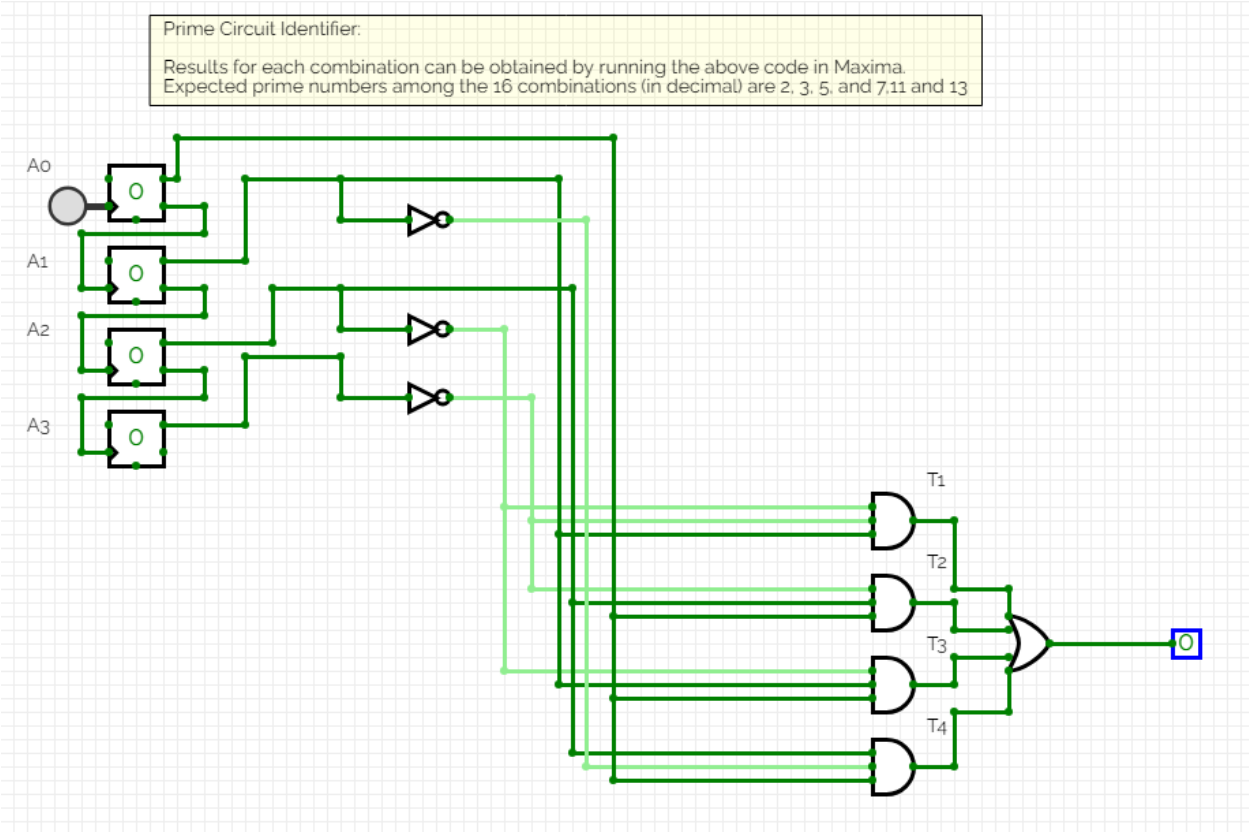
Results

Results for each combination can be obtained by running the above code in Maxima. Expected prime numbers among the 16 combinations (in decimal) are 2, 3, 5, and 7

Yellow Highlighted values from i8,i9,i11,i13,i17,i19 are prime numbers 2,3,5,7,11,13 respectively. Green Highlighted values are the Prime Equation Output

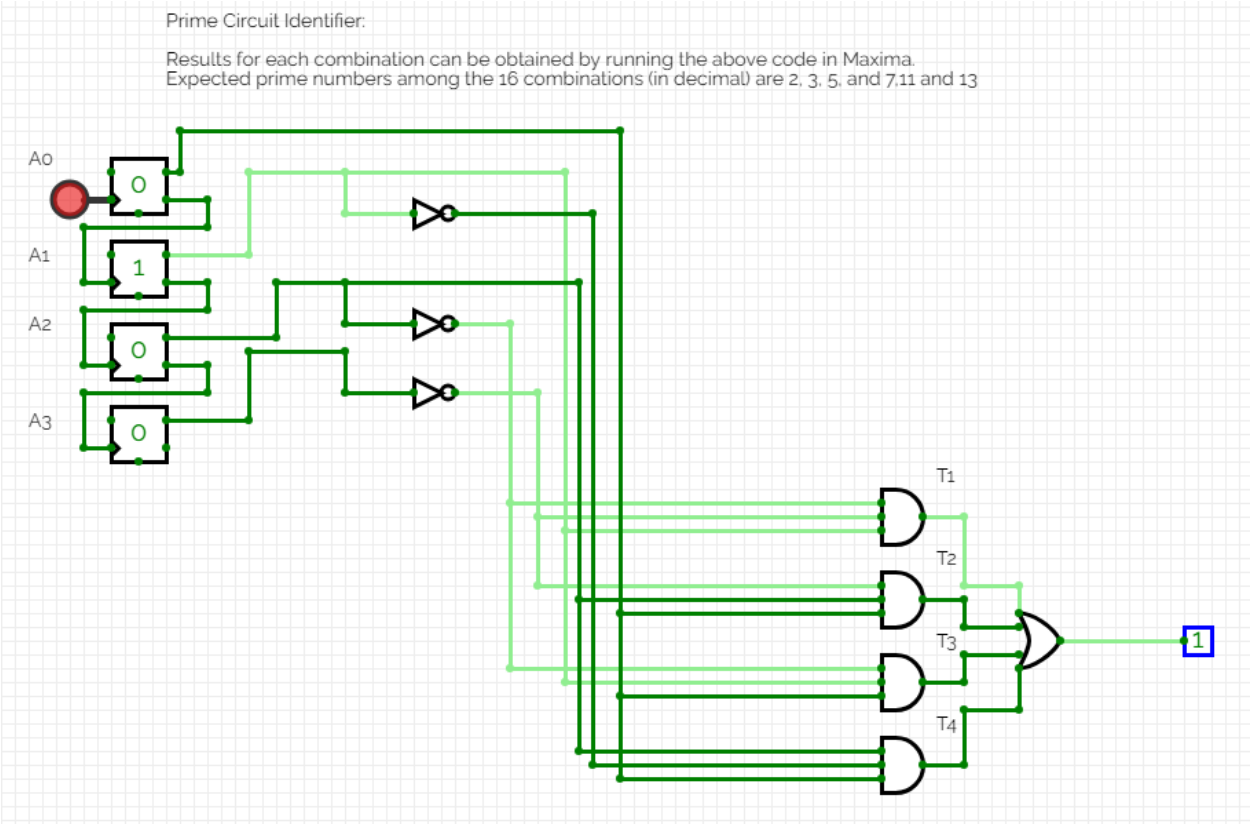
Circuit Diagram Screenshots:

This is the prime identifier circuit created in CircuitVerse

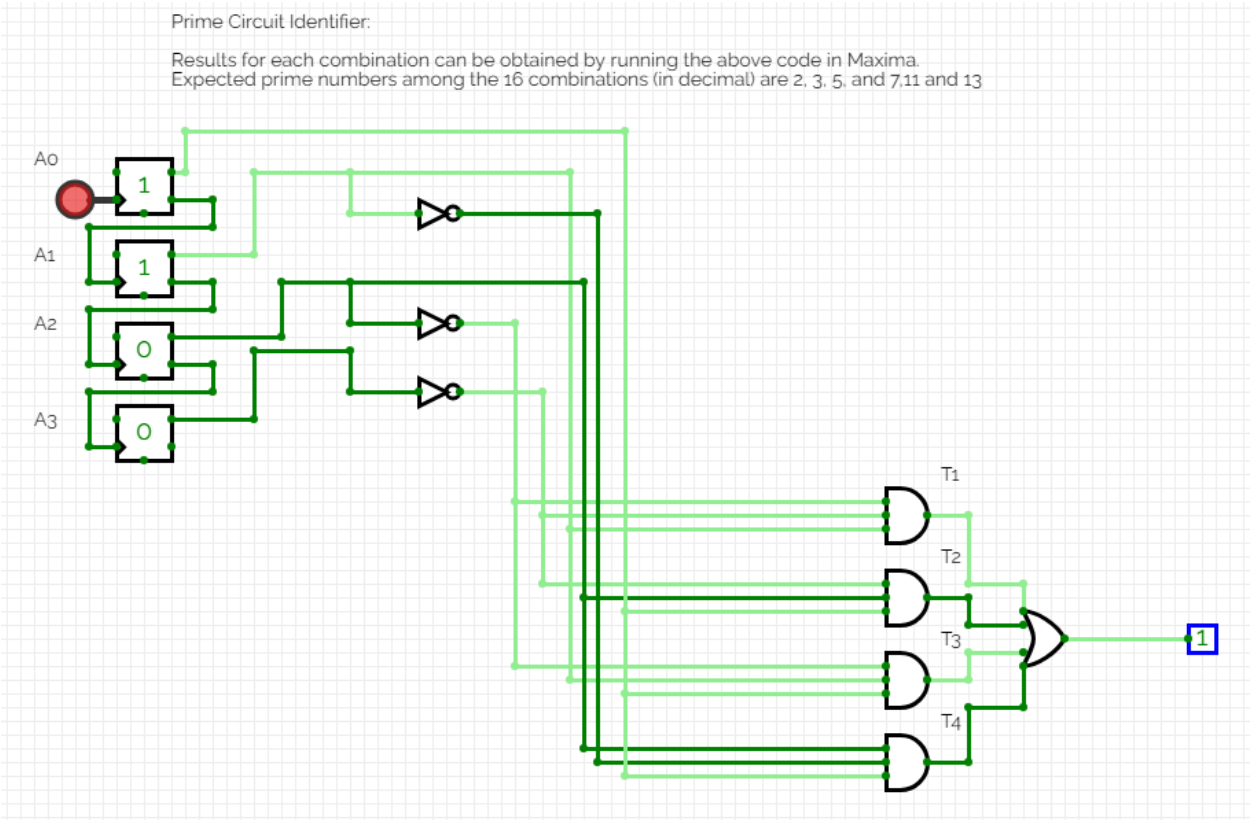


The following screenshots will show proof that Prime is true for the numbers that correspond to the Maxima output (which are 2,3,5,7,11,13):

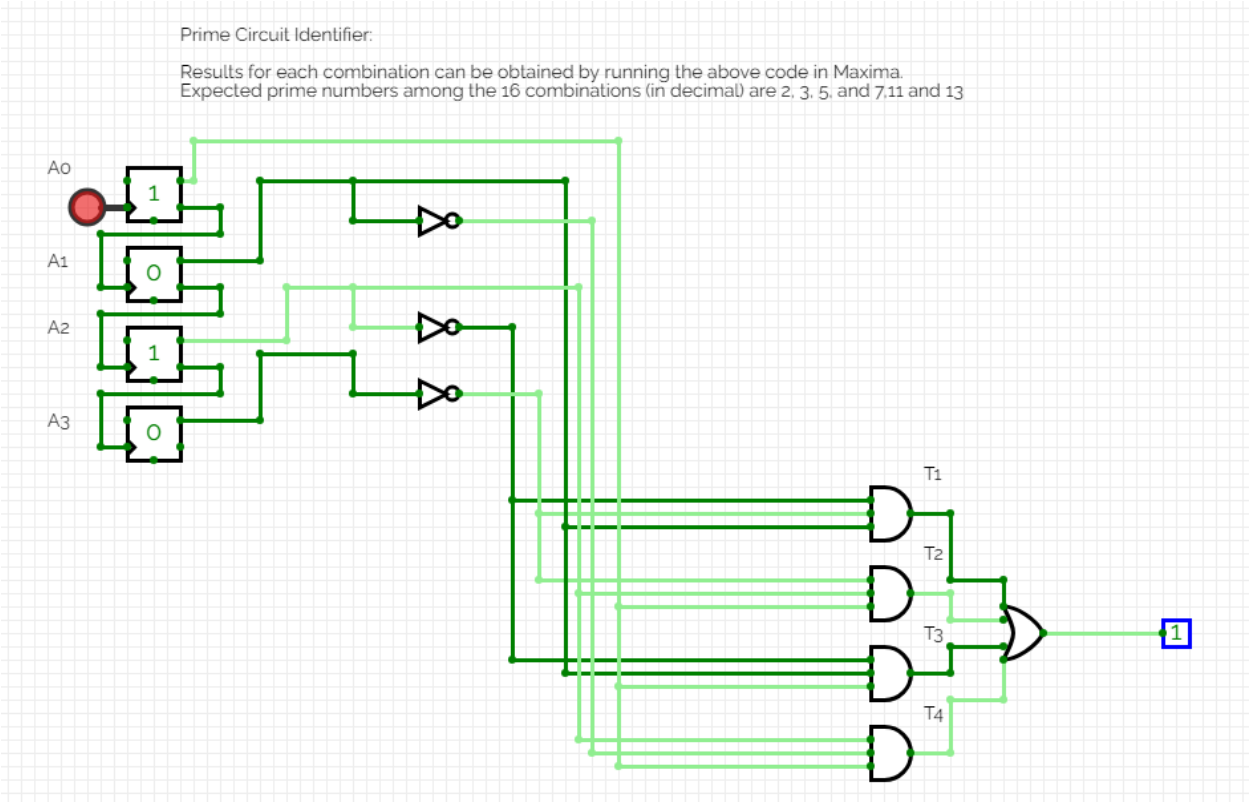
Circuitverse screenshot for Binary number 2 (expected 1):



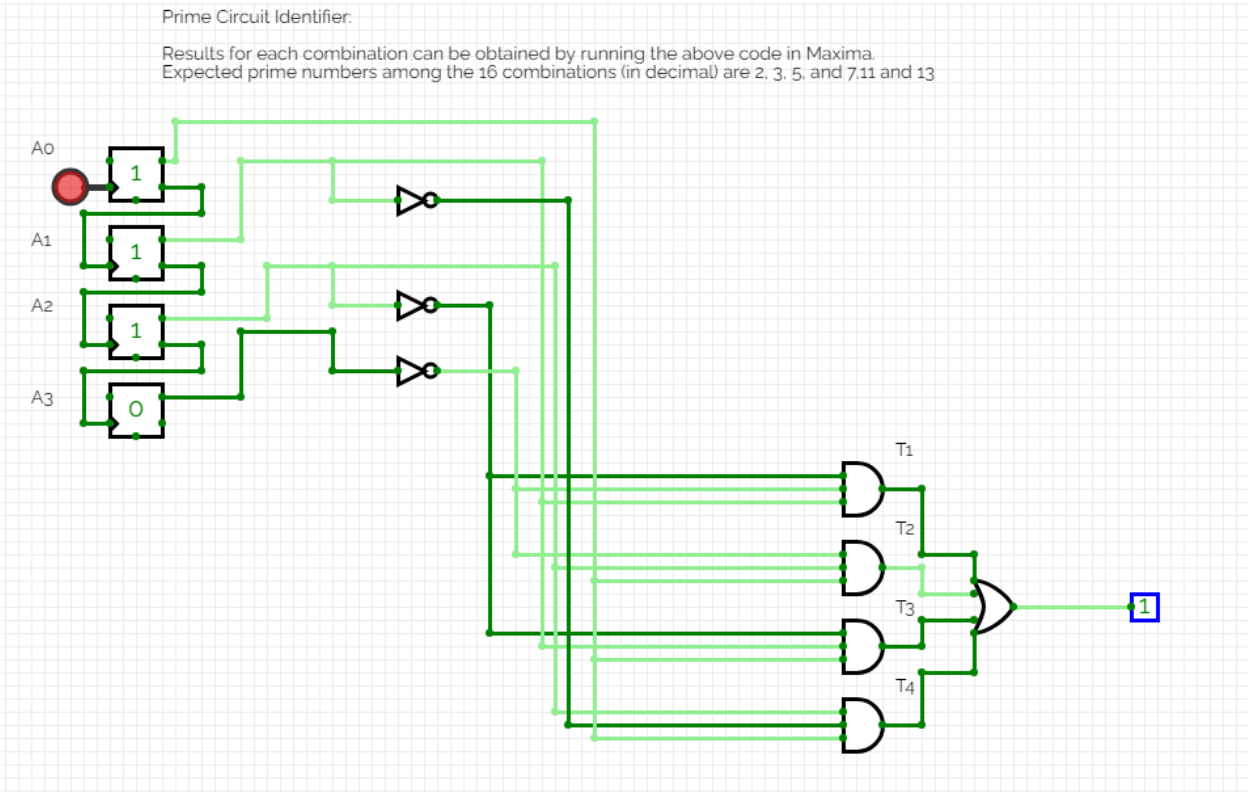
Circuitverse screenshot for Binary number 3 (expected 1):



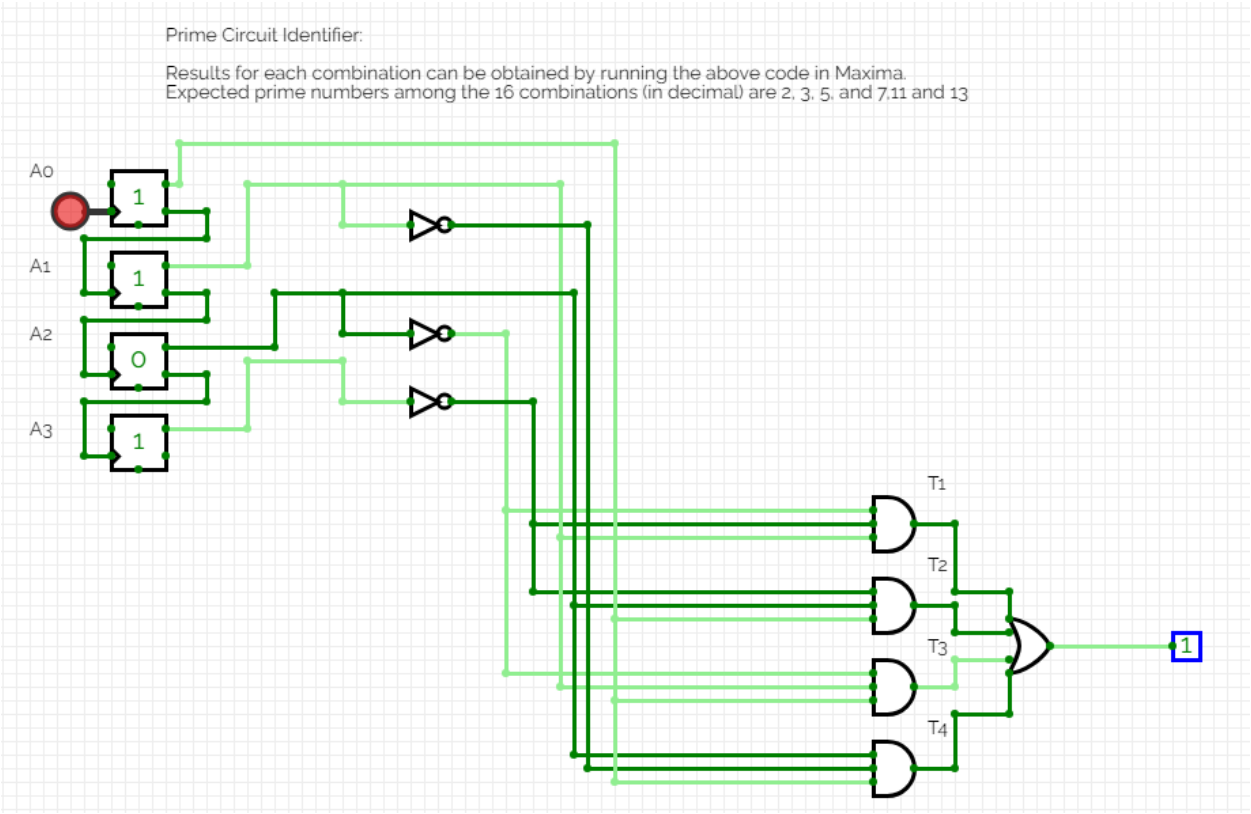
Circuitverse screenshot for Binary number 5 (expected 1):



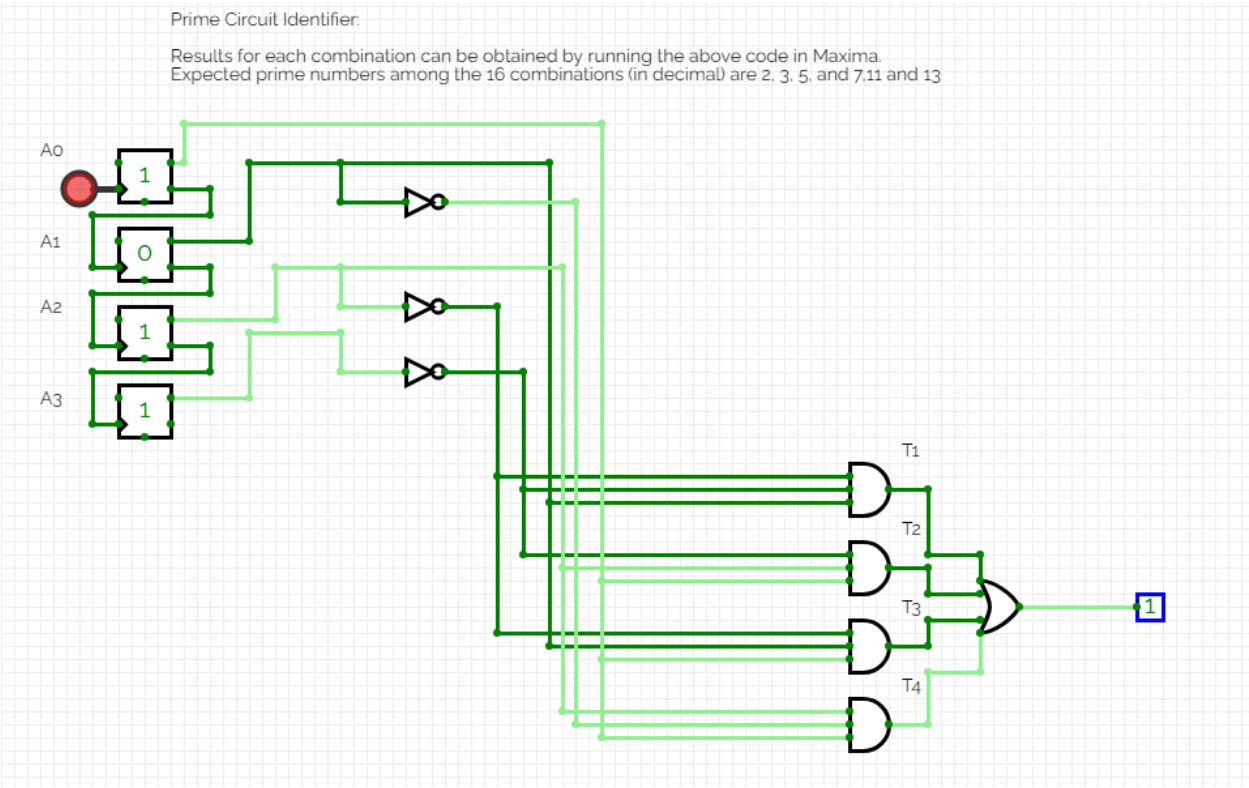
Circuitverse screenshot for Binary number 7 (expected 1):



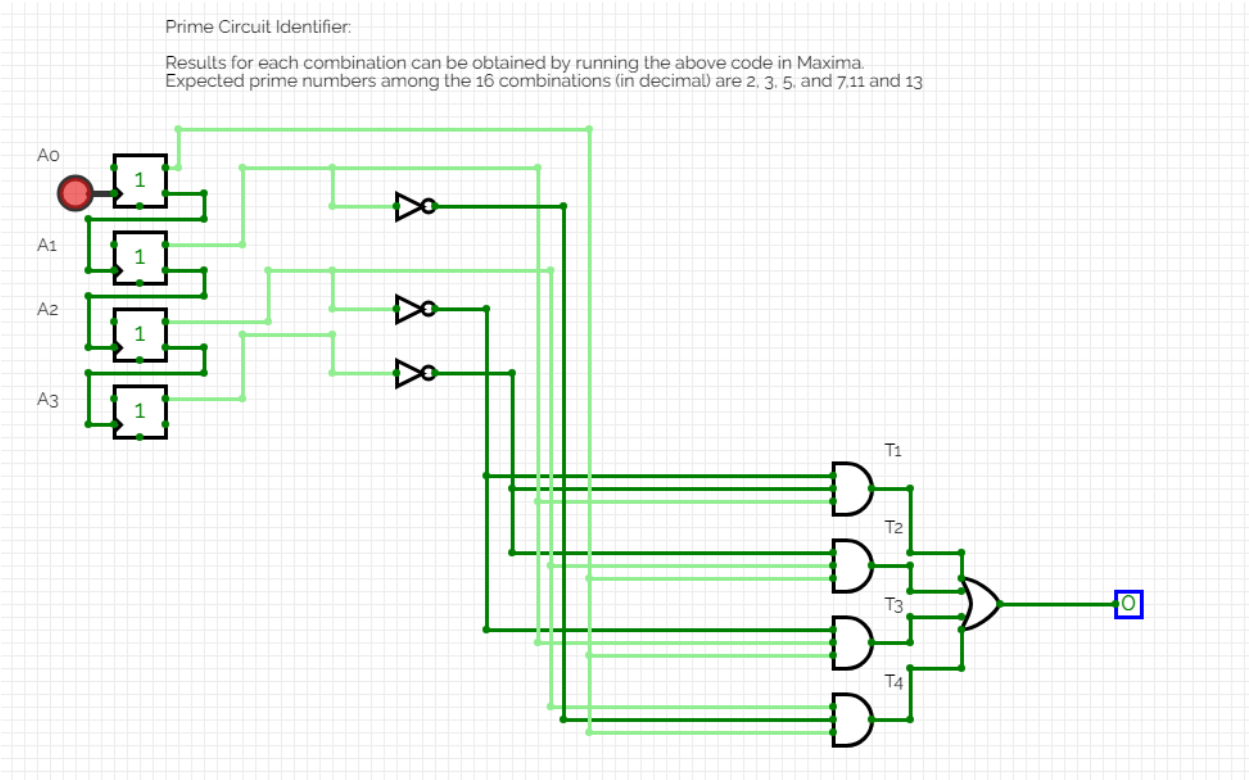
Circuitverse screenshot for Binary number 11 (expected 1):



Circuitverse screenshot for Binary number 13 (expected 1):



Circuitverse screenshot for Binary number 15 (expected 0): (INVALID PRIME)



Results:

We can see from the screenshot examples that the output was 1 for all binary representations of prime numbers and we can also see that the output is going to be 0 for all values that are not prime numbers in binary representation.

Conclusion

The simulation results from Circuitverse were compared to the expected outputs derived from the Maxima computations. It is evidenced that for each 4-bit input representing numbers 0 to 15, the Circuitverse simulation outputs precisely matched the Maxima-generated expected results. This confirmed the accuracy of the digital circuit design in identifying prime numbers. The exercise successfully demonstrates the application of Boolean algebra in digital circuit design and the effectiveness of simulation tools in verifying logical operations.