|  |
| --- |
| Name: Teh Jia Xuan Student ID: 32844700 |
| **FIT1047 Assignment 1: Numbers and Boolean Logic** |

Part 1

**Task 1.1 Representing numbers in binary**

**Advantages and Disadvantages of scientific notation**

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Able to perform multiplication with small values | rounding error may cause loss of information |
| Able to make additions with exponents | low accuracy as it could round off the large number |
| Able to represent tremendous values |  |

**Compare sign and magnitude, one’s complement and two’s complement.**

|  |  |  |
| --- | --- | --- |
|  | **Advantages** | **Disadvantages** |
| **i. Sign-magnitude** | -quickly identify positive and negative values.  -Doesn’t have overflows | -procedure would be tedious in addition  - Two zeros in the table (one of which is negative) |
|  |  |  |
| **ii. One’s complement** | -Able to do addition and subtraction easily | - Two zeros in the table (one of which is negative)  -Need to be cautious with overflows |
|  |  |  |
| **iii. Two’s complement** | -Solve the two zeros problem  -Negative 1 is always 111  -1000 is the smallest negative number  -Able to handle addition and subtraction easily | -Need to be cautious with overflows. |

One’s complement and sign magnitude have two zeros in their table, one of which is negative. Besides, the addition for sign-magnitude would be tedious compared to one’s and two’s complement. But sign-magnitude can quickly identify positive and negative values. On the other hand, One’s complement and two’s complement need to be careful of overflows, while sign magnitude doesn’t have this problem. Moreover, sign magnitude is divided two-part, the first bit would be the sign and the rest would be the magnitude. At the same time, one and two’s complements are treated as a whole. In conclusion, two’s complements have the best properties among all of them.

**Task 1.2 Converting to hexadecimal**

Student ID: 32844700

Negated Student ID: -32844700

**2) Converting negated value to 32-bit unsigned binary number**.

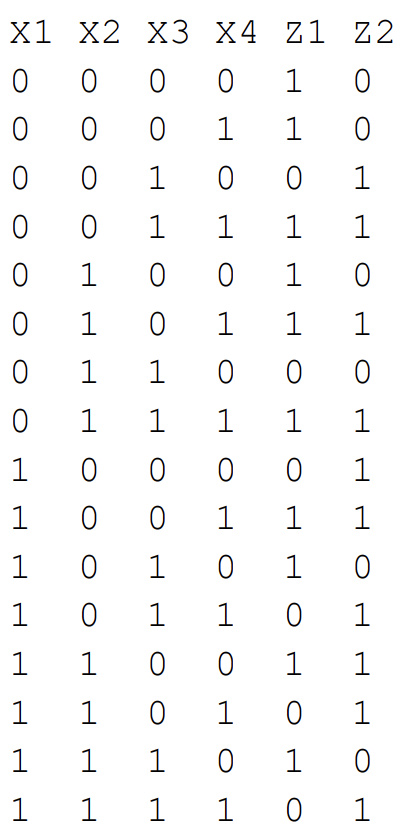
|  |  |  |
| --- | --- | --- |
| Base | Decimal | Remainder |
| 2 | 32844700 |  |
| 2 | 16422350 | 0 |
| 2 | 8211175 | 0 |
| 2 | 4105587 | 1 |
| 2 | 2052793 | 1 |
| 2 | 1026396 | 1 |
| 2 | 513198 | 0 |
| 2 | 256599 | 0 |
| 2 | 128299 | 1 |
| 2 | 64149 | 1 |
| 2 | 32074 | 1 |
| 2 | 16037 | 0 |
| 2 | 8018 | 1 |
| 2 | 4009 | 0 |
| 2 | 2004 | 1 |
| 2 | 1002 | 0 |
| 2 | 501 | 0 |
| 2 | 250 | 1 |
| 2 | 125 | 0 |
| 2 | 62 | 1 |
| 2 | 31 | 0 |
| 2 | 15 | 1 |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |
| 2 | 0 | 1 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Unsigned magnitude-32bits** | | 0000 0001 1111 0101 0010 1011 1001 11002 =  22+23+24+27+28+29+211+213+216+218+220+221+222+223+224 = 32844700 | | | | | | | |
| **Two’s complement-32bits** | | (Flipping all the bits from unsigned magnitude)  1111 1110 0000 1010 1101 0100 0110 00112  =1111 1110 0000 1010 1101 0100 0110 00112 + 1 (Add 1)  =1111 1110 0000 1010 1101 0100 0110 01002 = -32844700 | | | | | | | |
| **Two’s complement** | 11112 | | 11102 | 00002 | 10102 | 11012 | 01002 | 01102 | 01002 |
| **Hexadecimal Digit** | F | | E | 0 | A | D | 4 | 6 | 4 |

Hexadecimal notation: FE0AD46416

**Task 2.1 Boolean Algebra Expression**

**4 inputs X1,X2,X3,X4 and 2 output Z1, Z2 Truth table:**



**Boolean Function Z1 and Z2:**

Z1 (X1, X2,X3,X4) = + X4 + X3 X4 + X2 + X2 X4 + X2 X3 X4 + X1 X4 + X1 X3 + X1 X2 + X1 X2 X3

Z2(X1,X2,X3,X4) = X3 + X3 X4 + X2 X4 + X2 X3 X4 + X1 +

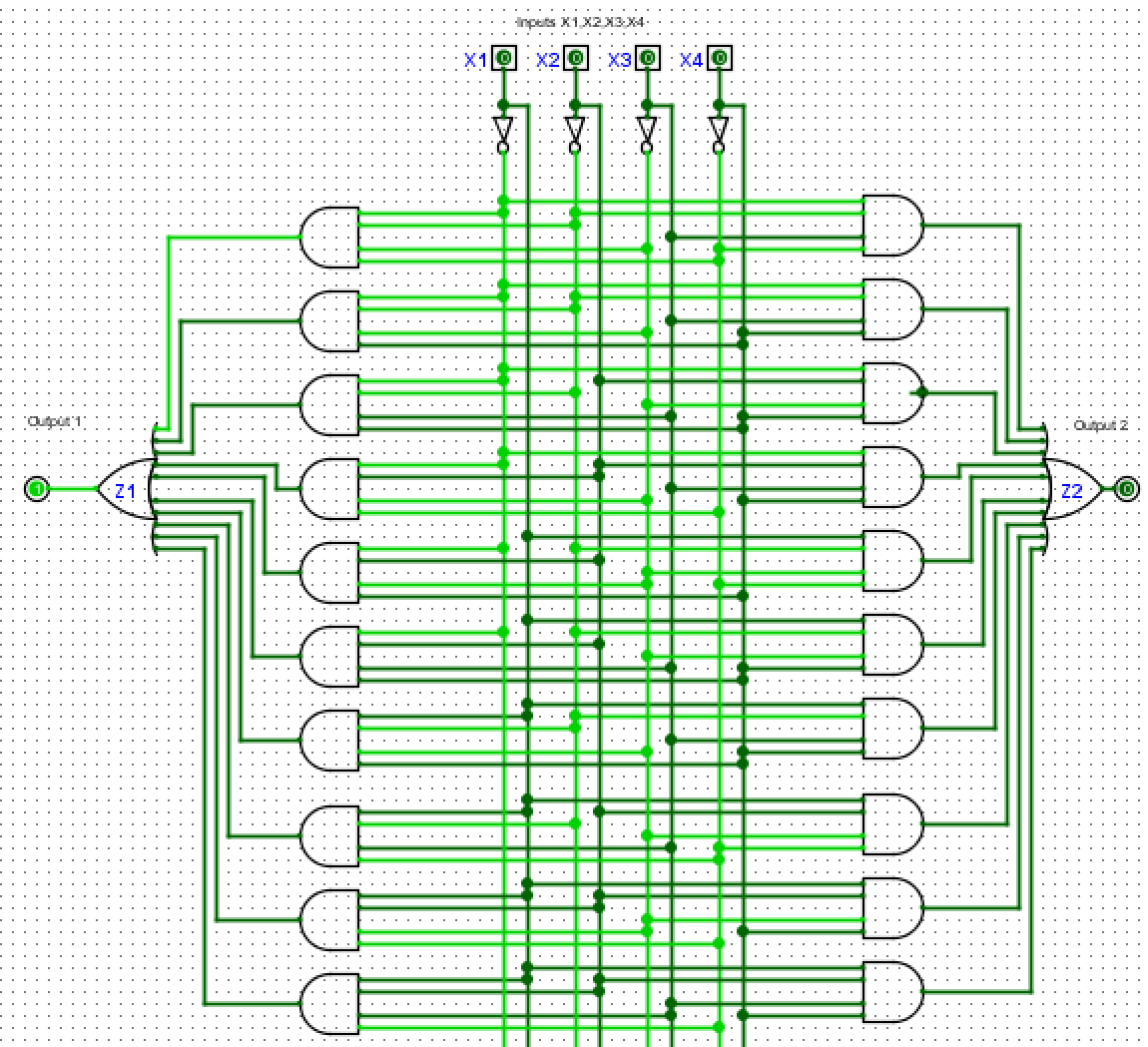
X1 X4 + X1 + X1 X2 + X1 X2 X4+ X1 X2 X3 X4

**Explanation:**

The rule sum of product is used to get the boolean functions for Z1 and Z2. There are 4 inputs which are X1 X2 X3 X4. 0 is defined as whereas 1 is defined as Xn where n is 1,2,3,4. The rows that returns 1 in Z1 and Z2 are chosen to form the boolean algebra. For example, Z1 returns 1 at row 1 of truth table. Thus, the inputs of X1 X2 X3 X4 are combined as to form a term as when the inputs of X1 X2 X3 X4 are 0 then Z1 returns 1. The same goes for the rows that return 1. After that, all terms are combined with “+” to form the boolean functions written above. Apart from that, the AND operation is used to connect the inputs X1,X2,X3,X4 to form terms according to the rows that return 1. While OR operation combines all the terms to create a function. Whereas NOT operation is use to make the inputs of .

**Task 2.2 Logical circuit in Logisim**

**Logisim Circuit Z1 and Z2 (Unsimplified version)**

****

**Explanation:**

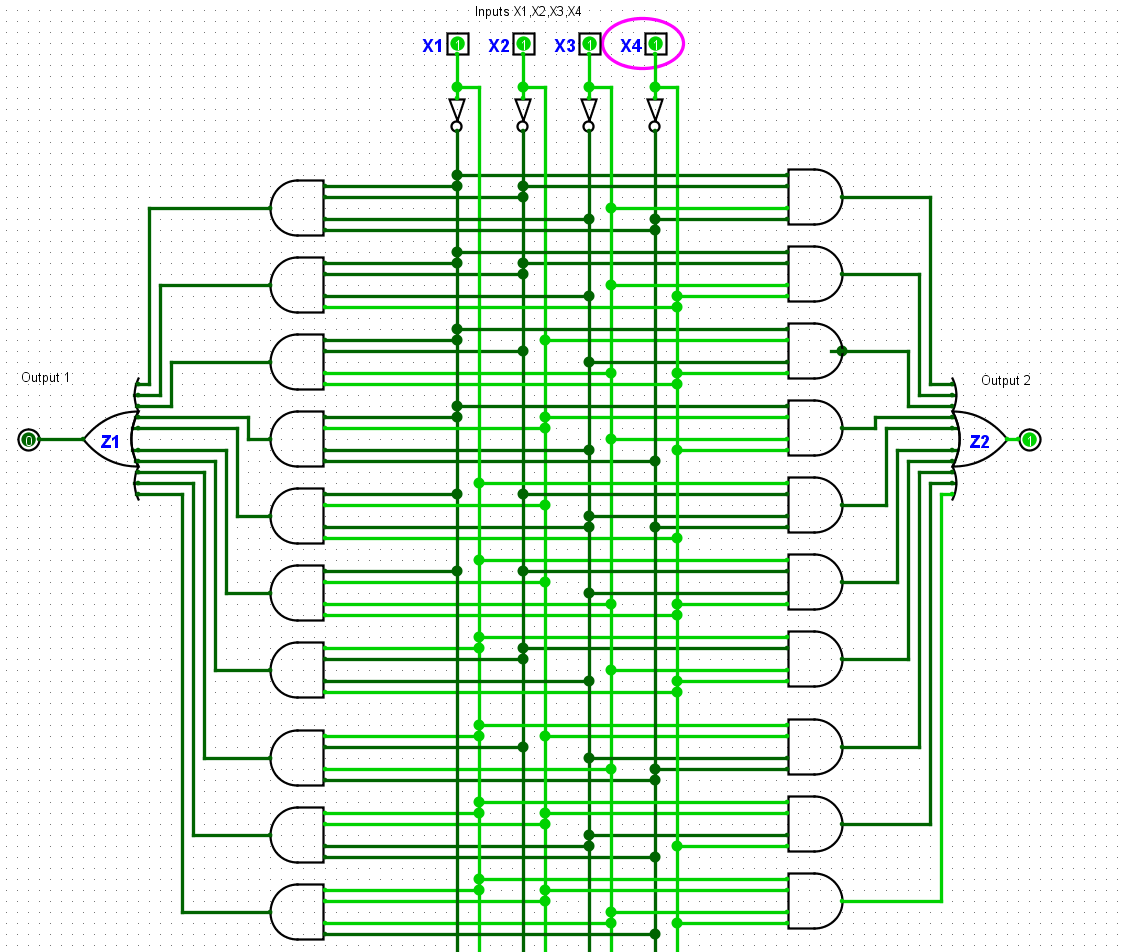
is defined as 0 Whereas X1 is defined as 1. Each inputs are connected to a NOT gate so that we can switch to 1 and 0 by connecting the wires to each. An AND gate is combined between the inputs by referencing the terms that are formed above. For example, will be connected to one AND gate and same goes to the other terms. After connecting all the inputs accordingly, all terms will be connected to an OR gate. For example, + X4 in order to combine these terms into a boolean function a “+” sign is used but in circuit we replace the “+” sign with an OR gate. Thus, all terms will be connected to an OR gate to form a complete circuit.

**The number of gates used**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Circuit** | **NOT gate** | **AND gate 4-inputs** | **OR gate 10-inputs** |
| **Z1** | 4 | 10 | 1 |
| **Z2** | 10 | 1 |

**Total** = 26 gates

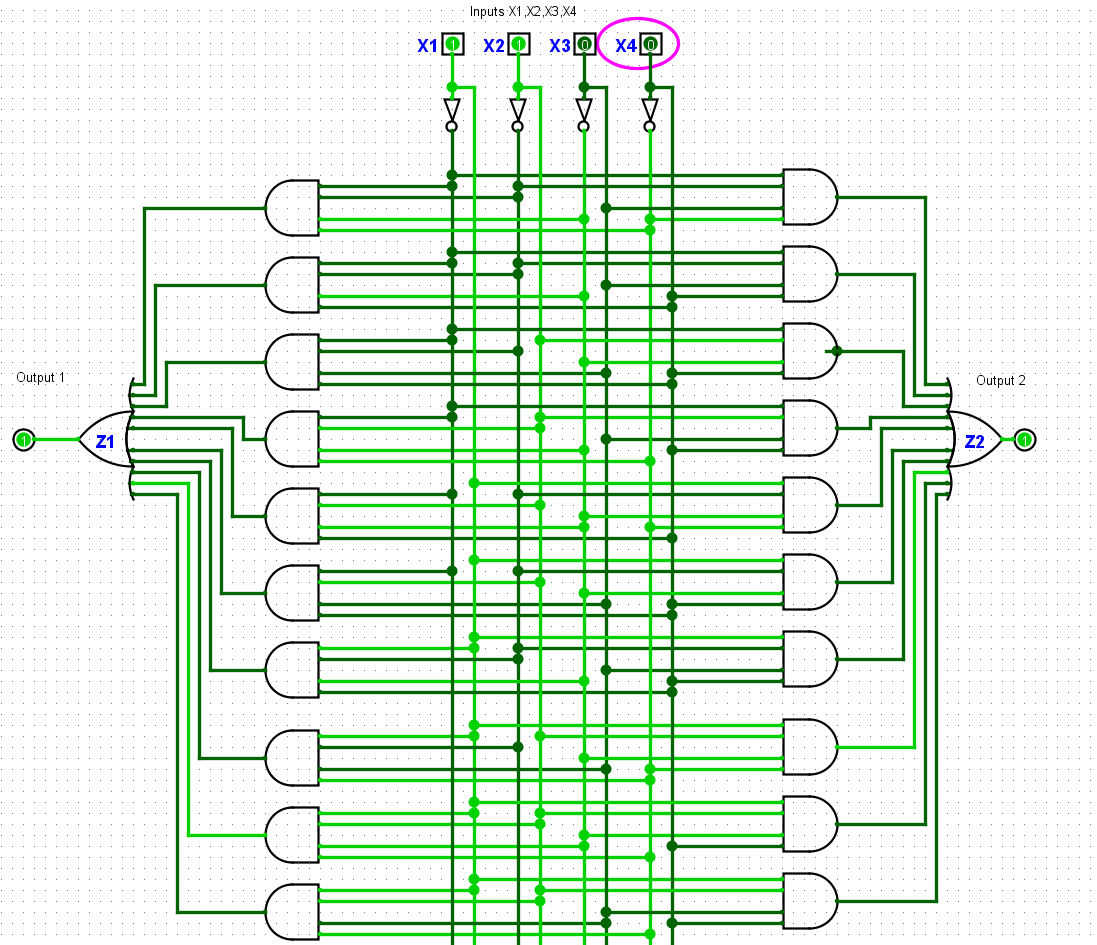
**Test Cases 1:**

****

Input Values: X1 = 1 ,X2=1, X3=1, X4=1

Output values = Z1 = 0 , Z2 = 1

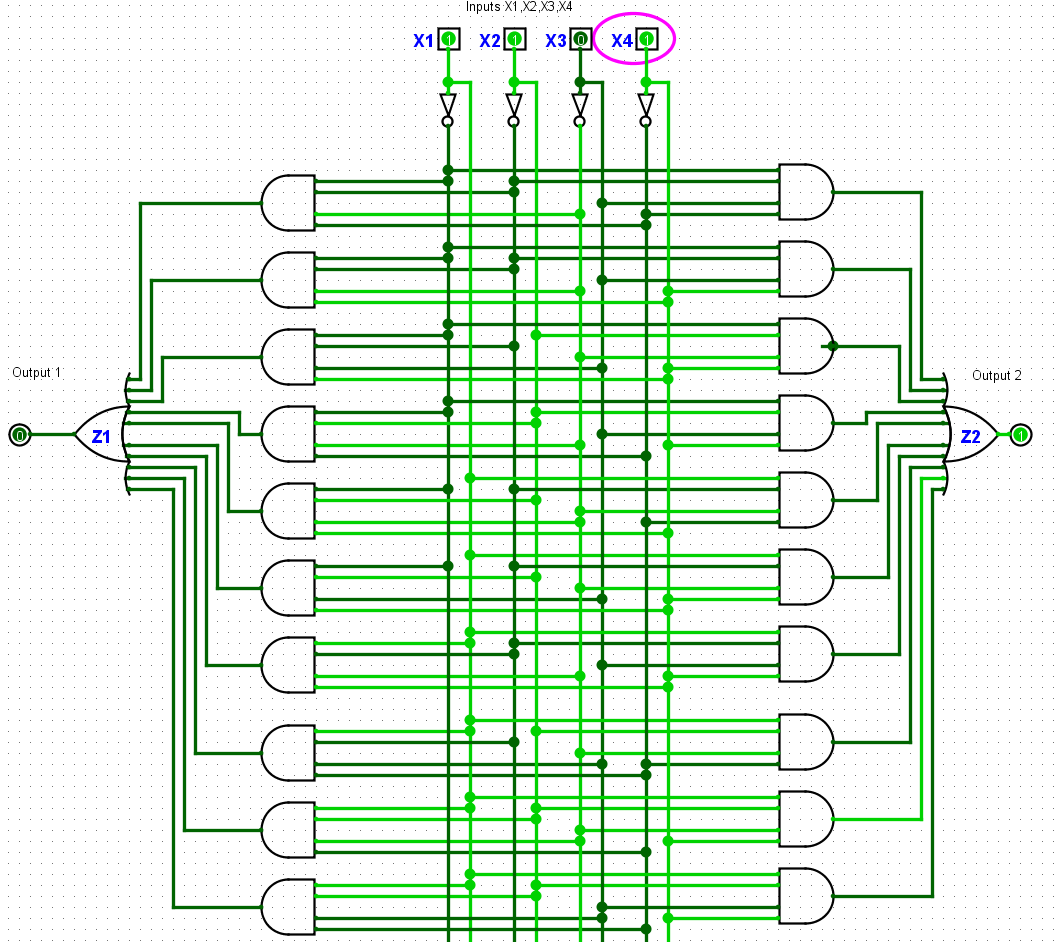
**Test Case 2**



Input values: X1 = 1, X2=1, X3= 0, X4=0

Output values : Z1 = 1, Z2=1

**Test Case 3**



Input values = X1 = 1, X2 = 1, X3 = 0, X4 = 1

Output values = Z1 = 0, Z2 = 1

**Task 2.3: Optimised circuit**

**Karnaugh map for Z1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X3x4**  **X1X2** | **0 0** | **0 1** | **1 1** | **1 0** |
| **0 0** | **1** | **1** | **1** | **0** |
| **0 1** | **1** | **1** | **1** | **0** |
| **1 1** | **1** | **0** | **0** | **1** |
| **1 0** | **0** | **1** | **0** | **1** |

**Optimised Z1 function =** + X2 X1 + X4 +X3 X4 + X3

**Karnaugh map for Z2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X3x4**  **X1X2** | **0 0** | **0 1** | **1 1** | **1 0** |
| **0 0** | **0** | **0** | **1** | **1** |
| **0 1** | **0** | **1** | **1** | **0** |
| **1 1** | **1** | **1** | **1** | **0** |
| **1 0** | **1** | **1** | **1** | **0** |

**Optimised Z2 function** = X1 + X2 X4 + X3 + X3 X4

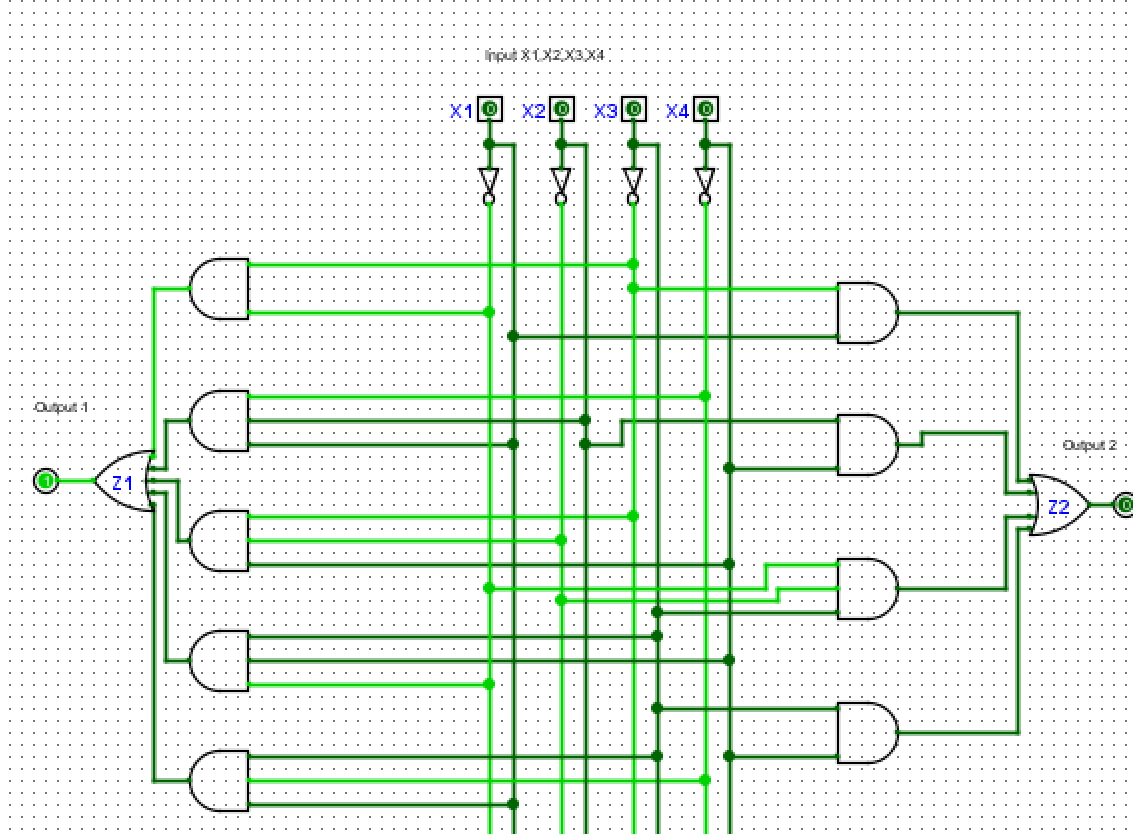
**Explanation:**

Construct a table above using the Sum of Products by following the truth table. Only groups of power of 2 can be grouped. Besides that, the value can wrap around the table if needed. And we need to take them as big group as possible with the power of 2.

By taking Z2 K-map as an example, 4 groups formed. For the group in blue colour, a group of 4 is formed as 4 is power of 2, and it is the biggest group that can be created. So after a group is formed, by looking at the label on top 0 0 0 1, the first 0 and third 0 represent X3 while the second 0 and fourth 1 represent X4 same goes for X1 and X2. The inputs of X2 and X4 don’t take into consideration as for the blue group Z2 always returns 1 regardless the inputs is 1 or 0. At the same time, inputs of X3 is 0 and X1 is 1. Both of these inputs remain the same in any combination. Thus, a X1 term is formed. The same goes for the rest of the groups

Lastly, we combined all the terms using OR operators to form a simplified boolean function.

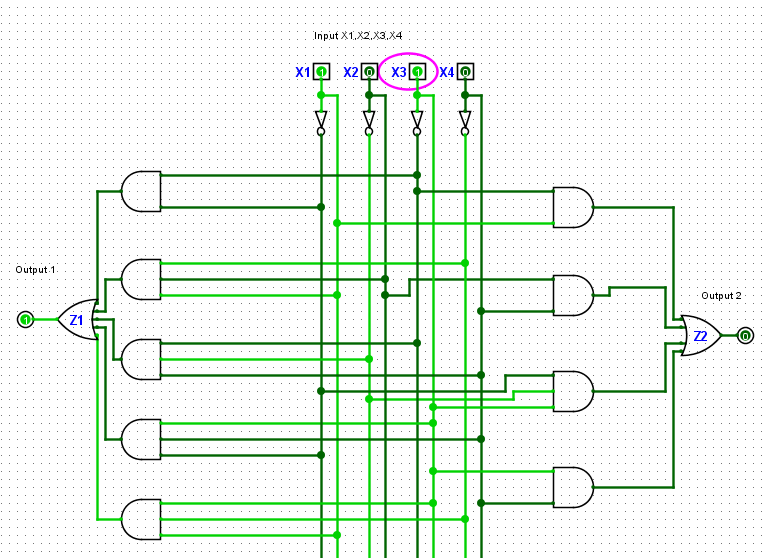
**Simplified Logisim Circuits Z1 and Z2**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Circuit** | **NOT gate** | **AND gate 4-inputs** | **OR gate 10-inputs** |
| **Z1** | 4 | 5 | 1 |
| **Z2** | 4 | 1 |

**Total = 15 gates**

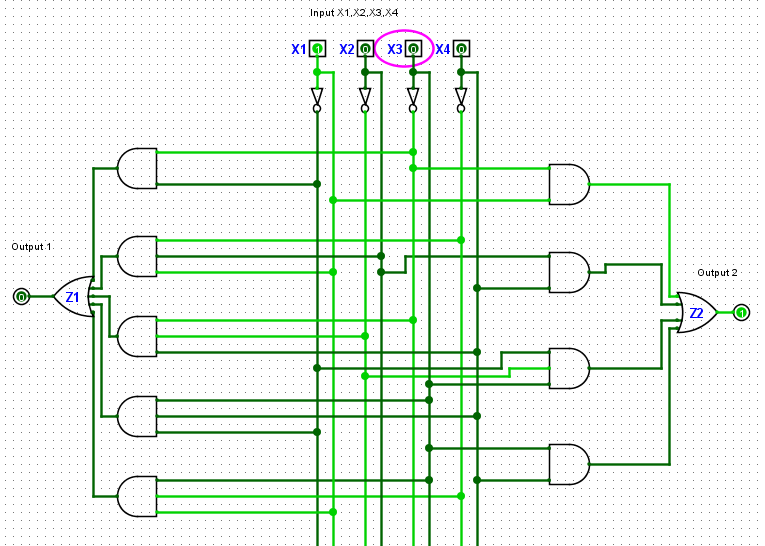
**Test Case 1:**

****

Input values = X1 = 1, X2 = 0, X3 = 1, X4 = 0

Output values = Z1 = 1 , Z2 = 0

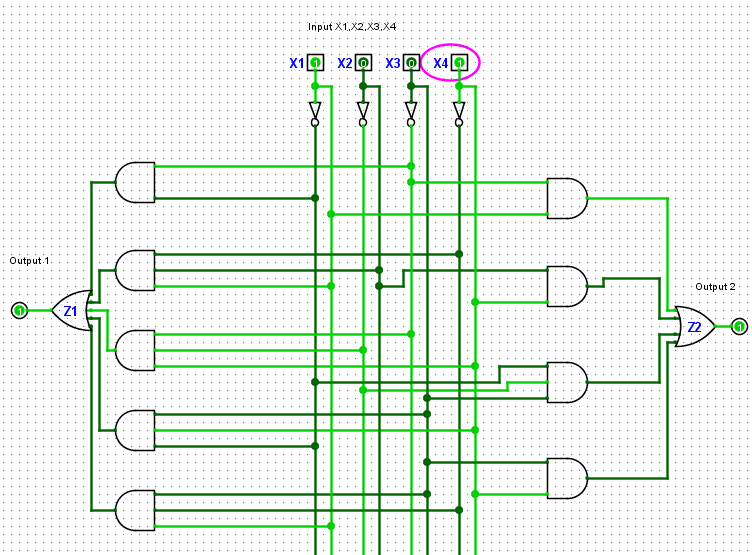
**Test case 2:**

****

Input values = X1 = 1, X2 = 0, X3 = 0, X4 = 0

Output values = Z1 = 0 , Z2 = 1

**Test case 3:**



Input values = X1 = 1, X2 = 0, X3 = 0, X4 = 1

Output values = Z1 = 1 , Z2 = 1