SW 1 - Binary To Ternary

Jonathan Schwartz

Section 003L

TA: Conner Rickermann

Date submitted: 2/12/2021

**Objective**

The purpose of this lab was to convert a number’s binary representation to a ternary representation of the form XYZ with each of these 3 terms stored as a 4-bit binary number. After finding these values for XYZ, the next step was to draft a table for the Z3,Z2,Z1,Z0 components, each of which is a binary bit, of the Z term of the ternary representation of the original number which was represented using binary bits D,C,B,A, before the conversion (The DCBA representation was in the binary number system). The reason for this lab is to gain practice with converting between, and to better understand multiple number systems. Once a table of these values was drafted, each binary bit of the Z term was given an equation in terms of D,C,B,A, and the compliments as well as the “and” (・) and “or” (+) operators, then these equations were to be used for forming a circuit in LogicWorks with binary witch inputs representing D,C,B, and A, and binary probes representing Z3, Z2, Z1, and Z0.

**Background/Introduction**

Converting between number systems is essential to computing, in communication between different circuits, and especially when used as an interface for a human. In certain areas of mathematics, converting between number systems is also incredibly important. Apart from simply converting between bases, the ternary number system as a whole is very useful too because it has the lowest radix economy due to being so close to the number “e” and is used in CMOS logic and transistor-transistor logic. Each number system is of the format xn\* bn + xn-1 \* bn-1 + … + x1 \* b1 + x0 \* b0, where x is the digit in the number, and b is the base that is used, so in ternary, the digits 0,1,2 are usable for x and b is 3 while in binary, only 0 and 1 are available for x and b is 2.

The method used to find the equation for the 4 different bits that define the Z term in ternary from the binary one is used often as a fundamental step in determining the logic of a circuit. Describing each bit’s equation as a function of the input bits (in this case 4) allows one to create a circuit of “and” and “or” gates which accounts for every case within the domain of the circuit. There are ways to consolidate the equations, but representing these conversions with “and” and “or” gates is essential for computing.

**Results**

The table below shows a theoretical circuit for the Z term in the binary to ternary conversion for the numbers 00002 to 10102 with labels for the most significant bit to the least significant bit from left to right (D - A and Z3 - Z0).

Theoretical Circuit - Binary to Ternary

|  | D | C | B | A |  | Z3 | Z2 | Z1 | Z0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input: | 0 | 0 | 0 | 0 | Output: | 0 | 0 | 0 | 0 |
| Input: | 0 | 0 | 0 | 1 | Output: | 0 | 0 | 0 | 1 |
| Input: | 0 | 0 | 1 | 0 | Output: | 0 | 0 | 1 | 0 |
| Input: | 0 | 0 | 1 | 1 | Output: | 0 | 0 | 0 | 0 |
| Input: | 0 | 1 | 0 | 0 | Output: | 0 | 0 | 0 | 1 |
| Input: | 0 | 1 | 0 | 1 | Output: | 0 | 0 | 1 | 0 |
| Input: | 0 | 1 | 1 | 0 | Output: | 0 | 0 | 0 | 0 |
| Input: | 0 | 1 | 1 | 1 | Output: | 0 | 0 | 0 | 1 |
| Input: | 1 | 0 | 0 | 0 | Output: | 0 | 0 | 1 | 0 |
| Input: | 1 | 0 | 0 | 1 | Output: | 0 | 0 | 0 | 0 |
| Input: | 1 | 0 | 1 | 0 | Output: | 0 | 0 | 0 | 1 |

Utilizing this table of theoretical values, it is possible to derive an equation for each bit in terms of D,C,B, and A as well as their compliments. The experimental equations are depicted below and will be used in creating the circuit.

**Z3: 0**

**Z2: 0**

**Z1:** D’C’BA’ + D’CB’A + DC’B’A’

**Z0:** D’C’B’A + D’CB’A’ + D’CBA + DC’BA’

After creating the circuit and testing each input depicted above, the experimental values matched the theoretical values for each case. The table below shows the data obtained from the circuit built off of the above equations.

Experimental Circuit - Binary to Ternary

|  | D | C | B | A |  | Z3 | Z2 | Z1 | Z0 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input: | 0 | 0 | 0 | 0 | Output: | 0 | 0 | 0 | 0 |
| Input: | 0 | 0 | 0 | 1 | Output: | 0 | 0 | 0 | 1 |
| Input: | 0 | 0 | 1 | 0 | Output: | 0 | 0 | 1 | 0 |
| Input: | 0 | 0 | 1 | 1 | Output: | 0 | 0 | 0 | 0 |
| Input: | 0 | 1 | 0 | 0 | Output: | 0 | 0 | 0 | 1 |
| Input: | 0 | 1 | 0 | 1 | Output: | 0 | 0 | 1 | 0 |
| Input: | 0 | 1 | 1 | 0 | Output: | 0 | 0 | 0 | 0 |
| Input: | 0 | 1 | 1 | 1 | Output: | 0 | 0 | 0 | 1 |
| Input: | 1 | 0 | 0 | 0 | Output: | 0 | 0 | 1 | 0 |
| Input: | 1 | 0 | 0 | 1 | Output: | 0 | 0 | 0 | 0 |
| Input: | 1 | 0 | 1 | 0 | Output: | 0 | 0 | 0 | 1 |

**Discussion/Results**

The experimental results may have matched the theoretical results, however the circuit which was used has a lot of excess parts and very few wires were used multiple times. There are ways to abstract certain parts such as if there is a D’ component in each of two terms within an equation for a bit, it can be abstracted and used once rather than twice. The gates used were also limited to “2” gates for “and” and “or” operators, which required many more gates when 4 wires are compared in each term.

Even though the circuit was inefficient, it still did what was expected and displayed the correct result, so it was not a bad circuit, it could simply be improved upon.

**Questions**

“What do you think you could do to reduce the number of gates in your solution?”

As mentioned in the discussion section, using “and-4” and “or-3” or “or-4” gates would reduce the total number of gates, but also there is some level of redundancy in the equations we developed in this lab, and using the method where certain common factors can be used once rather than many times over would reduce the number of gates required to compare all of them.