Implementation of Boolean Algebra and Binary Arithmetics in Constructing A Redstone Based Calculator in Minecraft

Rifqi Irfan Indrawan - 13525061
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: rifqiirfanindrawan@gmail.com, 13525061@std.stei.itb.ac.id

***Abstract*—Boolean algebra and binary arithmetic form the theoretical foundation of all modern digital computation, yet these concepts often remain abstract within a purely mathematical context. This paper presents the design and implementation of a 8-bit binary calculator capable of performing addition and subtraction operations, realized entirely through the Redstone mechanism in Minecraft 1.18.2 as a tangible demonstration of discrete mathematics principles.**

Keywords—Boolean Algebra, Binary Arithmetics, BCD Display, Binary Decoder, Minecraft, Calculator

I. Introduction

Modern computers at its core use a complex system to efficiently execute an action. This is done to efficiently use the computer's Core Processing Unit (CPU). However, we as humans tend to have difficulties understanding how a computer works. For example, how does a computer compute numbers? It is not as simple as we humans have done it since elementary.

Calculations are an essential part of digital computational systems. A big part of said system is boolean algebra, a branch of algebra where values of variables are divided into 2 values, true (1) and false (0), different from elementary algebra where values of variables are numbers. Another part of the system are binary arithmetics, where binary is a base-2 number system that uses 2 states, either a 1 or a 0, and binary arithmetic is an essential part of the digital computational system that can be used to implement addition, subtraction, multiplication, and division using binary.

To better understand the use and implementation of these theories, there are multiple simulations, one of them is Minecraft, a sandbox game that can simulate calculations using redstone systems similar to computers.

II. Theory

II.A. Number Systems

Number systems are a method to represent numbers with a set of rules and symbols. Since computers can only understand two conditions, ON and OFF, they are represented as 1 and 0 respectively, this is also called base-2. This is different from how humans perceive numbers, base-10 or decimals. For example, if we ask a computer and a human to represent 13 it would look like this;

$$13_{10} = 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 1101_2$$

In addition, bits are individual 1s or 0s inside a number, nibbles 4 bits, and bytes are 8 bits. Other number systems such as octals and hexadecimals will not be discussed in this paper but are a part of number systems as a whole.

II.B. Boolean Algebra & Logic Gates

Boolean Algebra is a branch of algebra that is directly connected to the use of binary where their values are represented as either True for 1s or False for 0s. It has 3 main operators, AND ($\wedge$), OR ($\vee$), dan NOT ($\neg$). This system was introduced in 1854 by George Boole and has become the foundation of digital circuit designs. The core foundation of boolean algebra are based on its laws, such as;

Identity Law:

1. $a + 0 = a$
2. $a \cdot 0 = 0$

Complement Law:

1. $a + a' = 1$
2. $a \cdot a' = 0$

Idempotent Law:

1. $a + a = a$
2. $a \cdot a = a$

De Morgan Law:

1. $(a \cdot b)' = a' + b'$
2. $(a + b)' = a' \cdot b'$

Absorption Law:

1. $a + a \cdot b = a$

In Boolean Algebra, a boolean function is a standard way to represent an entire rule. The two standard ways of representing a boolean function are through its canonical forms, Product Of Sum or Sum Of Product.

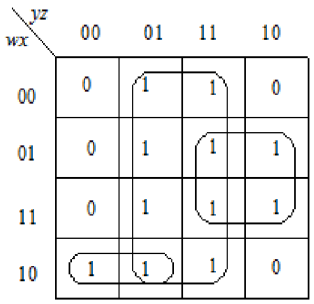
$$h(a, b, c) = abc'$$

The above function, h, states that it has 3 literals, a, b, and c'.

The canonical forms of a boolean function states that a function can be represented in 2 different ways, Product Of Sum (POS) or Sum Of Product (SOP). SOP uses a chain of ORs of its minterm, and POS uses a chain of ANDs of its maxterm.

In terms of boolean functions, one way to minimize and visualize functions is to utilize Karnaugh Maps (K-Map), introduced by Maurice Karnaugh in 1953. K-Maps are used to simplify functions through a table. We generally need to minimize functions to save space and resources.

Such a way to minimize a 4 literals functions are as follows, the first step is to fill up the K-Map with 1s using a truth table that precedingly have been generated and/or made. The second step is to fill the rest of the map with 0s. The third step is to wrap around with the order of octet, followed by quads, then finally duplets.



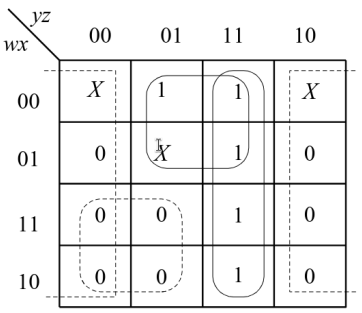
Hasil penyederhanaan: $F(w, x, y, z) = z + xy + wx'y'$

TABLE I

EXAMPLE OF A 4-VARIABLE K-MAP[13]

In some cases, there would be a *don't care* condition. These conditions appear when there are values that we don't necessarily need. When doing a simplification using *don't care* conditions, they can be interpreted as either a 1 or a 0 but not both at the same time.

an:



Hasil penyederhanaan:
SOP: $F(w, x, y, z) = yz + w'z$ (kelompok garis penuh)
POS: $F(w, x, y, z) = z\,(w' + y)$ (kelompok garis putus-putus)

TABLE II

EXAMPLE OF DON'T CARE CONDITIONS IN A K-MAP[13]

Lastly, functions in Boolean Algebra can also be represented as logical gates.

Name	Graphical Symbol	Algebraic Function	Truth Table
AND	A, B → F	F = A • B or F = AB	A B | F: 0 0 | 0; 0 1 | 0; 1 0 | 0; 1 1 | 1
OR	A, B → F	F = A + B	A B | F: 0 0 | 0; 0 1 | 1; 1 0 | 1; 1 1 | 1
NOT	A → F	$F = \overline{A}$ or F = A′	A | F: 0 | 1; 1 | 0
NAND	A, B → F	$F = \overline{AB}$	A B | F: 0 0 | 1; 0 1 | 1; 1 0 | 1; 1 1 | 0
NOR	A, B → F	$F = \overline{A + B}$	A B | F: 0 0 | 1; 0 1 | 0; 1 0 | 0; 1 1 | 0
XOR	A, B → F	$F = A \oplus B$	A B | F: 0 0 | 0; 0 1 | 1; 1 0 | 1; 1 1 | 0

Fig. 1. Basic logic gates and their Boolean representations: AND, OR, NOT, NAND, NOR, and XOR. [14]

II.C. Binary Arithmetics

Binary Arithmetic is an essential part of digital systems. Within it, computers can do basic arithmetic operations such as addition, subtraction, multiplication, and division using binary. These systems use a heavily efficient way to do operations to maximize its efficiency and space.

Additions are done using a combination of half adders and full adders and finally using said combination for a 4 bit Ripple Carry Adder.

1. Half Adder

A half adder is used for adding 2 bits together, resulting in 2 outputs, Sum (S) and Carry (C). The sum is done by XORing both A and B (the inputs). While the carry is done by the product of both A and B (AND).

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

TABLE III

TRUTH TABLE FOR HALF ADDER[8]

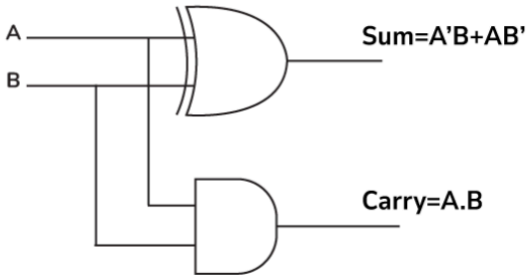


Fig. 2. Logic Gate for Half Adder[8]

$$S = A \oplus B$$

$$C = AB$$

2. Full Adder

Compared to a half adder, a full adder is much more complex, it allows for 3 bits instead of 2. The sum is also done by XORing all 3 inputs. Meanwhile the carry is done by the sum of product of two inputs.

A	B	C	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	0
1	1	1	1	1

TABLE IV

TRUTH TABLE FOR FULL ADDER[8]

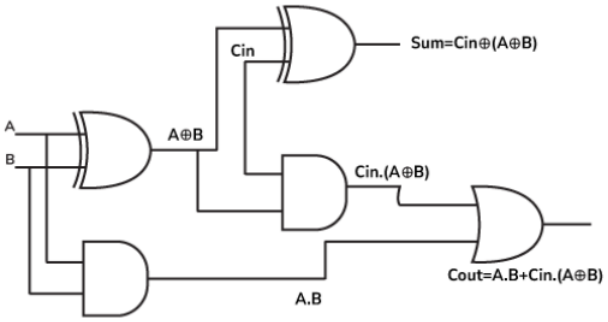


Fig. 3. Logic Gate for Full Adder[8]

$$S = A \oplus B \oplus C$$

$$C = AB + BC + AC$$

3. 8 bit Ripple Carry Adder

A Ripple Carry Adder is a type of full adder that is arranged in a way that each carry outs "ripples" through one another, creating a chain of full adders. A 8 bit Ripple Carry Adder means that it uses 8 full adders stacked together to create a maximum binary of 8 bits. The limitations of this would be that the range of values it can produce are $\{0, 255\}$

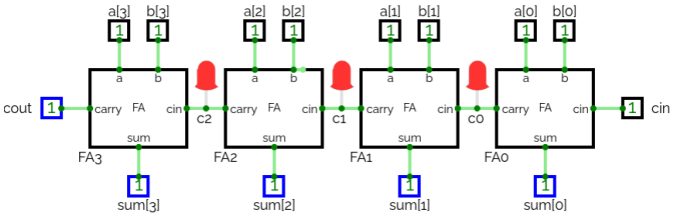


Fig. 4. Logic Gate for a 4 Bit Ripple Carry Adder [11]

For example,

$$5_{10} + 3_{10} = 0101_2 + 0011_2$$

$$1 + 1 = 0, C_1 = 1 \rightarrow S_0 = 0$$

$$0 + 1 + 1 = 0, C_2 = 1 \rightarrow S_1 = 0$$

$$1 + 0 + 1 = 0, C_3 = 1 \rightarrow S_2 = 0$$

$$0 + 0 + 1 = 1, C_4 = 0 \rightarrow S_3 = 1$$

$$S_0 + S_1 + S_2 + S_3 = 0 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3$$

$$S_0 + S_1 + S_2 + S_3 = 8_{10} = 1000_2$$

Subtractions are done using a representation of negative numbers in binary, called Two's Complement. Instead of using another operator, subtractions just use addition but with its Two's Complement, meaning,

$$A - B = A + (-B)$$

Where $-B$ is the Two's Complement of B. To get that value, we need to flip all of B's bits, and then add 1 to the flipped result.

$$-B = \sim B + 1$$

In terms of the ripple carry adder, $+1$ is the C_0 of the first full adder, or the C_{in}. For example,

$$7_{10} - 3_{10} = 0111_2 - 0011_2 = 0111_2 + \sim(0011_2) + 1$$

$$\sim 0011_2 = 1100_2$$

$$1 + 0 + 1 = 0,\ C_1 = 1 \rightarrow S_0 = 0$$

$$1 + 0 + 1 = 0,\ C_2 = 1 \rightarrow S_1 = 0$$

$$1 + 1 + 1 = 1,\ C_3 = 1 \rightarrow S_2 = 1$$

$$0 + 1 + 1 = 0,\ C_4 = 1 \rightarrow S_3 = 0$$

$$S_0 + S_1 + S_2 + S_3 = 0 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 0 \times$$

$$S_0 + S_1 + S_2 + S_3 = 4_{10} = 0010_2$$

Since the subtraction is also using the 8-bit Ripple Carry Adder, the limitations are the same, the range of the values it can produce are $\{-128, 127\}$

II.D. BCD Decoder to 7-Segment Display

Binary Coded Decimal (BCD) is a method to transform decimals into binary. A BCD Decoder is a system designed to decode said binary into sets of rules for a 7-Segment Display. These sets of rules mainly apply the use of K-Maps

7-Segment Displays are a way to output binaries into a device that will visualize said binary into 7 different segments. The BCD Decoder will map its outputs to the 7-Segment Display to later be outputted as their own decimal numbers.

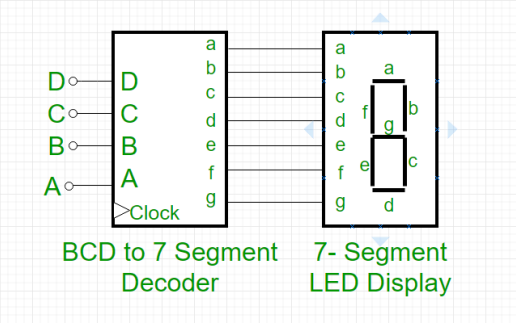


Fig. 5. BCD to 7 Segment Decoder and its Display Mapping [3]

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	Don't Care
11	Don't Care
12	Don't Care
13	Don't Care
14	Don't Care
15	Don't Care

TABLE V

BCD REPRESENTATION OF DECIMAL DIGITS 0–15

II.E. Logic Gates Using Redstone

Redstone is an in-game material used in Minecraft to simulate electrical signal behavior, which can be used as a medium for implementing logical circuits/machinery within its environment. As mentioned in the Boolean Algebra section, there are 3 core components, this is the mapping from Minecraft redstone to logical components are as follows,

Redstone Component	Behavior	Logic Equivalent
Redstone dust	Carries signal (ON/OFF)	Wire
Redstone torch	Outputs ON when input is OFF	NOT gate
Redstone repeater	Passes and delays signal	Buffer/signal restorer
Redstone comparator	IF gate or INHIBIT in Comparison Mode, XOR or XNOR in Subtract Mode	XOR
Lever/button	Manual ON/OFF	Binary Input (1/0)
Redstone lamp	Lights up when received signal	Output

TABLE VI

MAPPING OF REDSTONE COMPONENTS TO LOGIC EQUIVALENTS

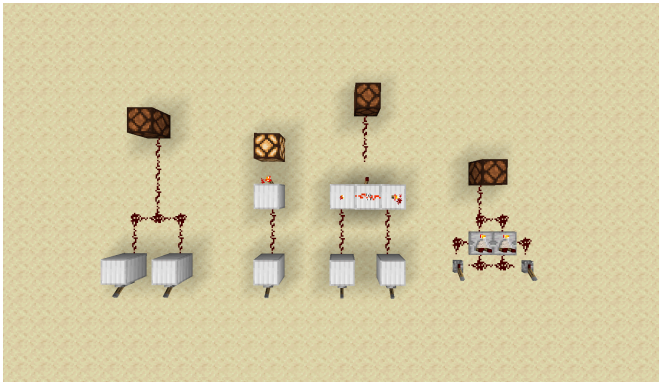


Fig. 6. Core Logic Gates Implemented in Minecraft

For each of the core logic gates, here are the implementation of OR, NOT, AND, and XOR gates. In the case of redstone, everything happens discretely in game ticks (1 tick = 0.1 seconds) this is important to repeaters because they operate in deliberate delays, which will affect the carry signals propagating through multiple full adders in a sequence.

While Minecraft is running on a game environment, redstone's ON/OFF behavior is mathematically equivalent to Boolean logic, making it a suitable environment for this type of contraption.

III. Methodology And Implementation

III.A. System Design

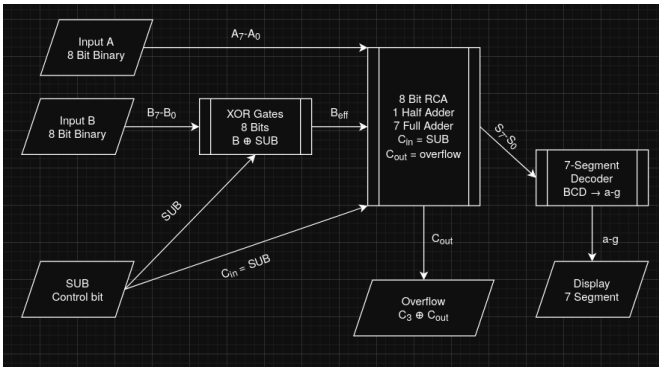


Fig. 7. Block diagram of the 8-bit Redstone calculator system showing input, processing, and output stages.

Overall, the entire system could be described in this Diagram Block. It starts out from the Input blocks, then to processing, and finally the outputs and displays. The input blocks are taken from the keypad in the final build, plus the SUB input to control if the calculation would be an addition or a subtraction. The processing blocks are the XOR gates for Two's Complement, the 8-Bit Ripple Carry Adder itself, and the BCD to 7-Segment Decoder. Finally, the output blocks are the overflow and the 7-Segment display.

The input stage as described before is taken from toggling levers and pressing a keypad in the system. The user provides 2 8-bit numbers, A and B, and 1 SUB to determine the operation done to said numbers. The SUB bit performs 2 things simultaneously, it feeds into all the XOR gates on the B input line and sets the carry-in of the first adder stage to 1. This is done to efficiently set the Two's Complement on hand whenever the user wants subtraction, but also to let B's input fall through to the next process whilst doing the XOR stage.

The processing stage receives the inputs of A, B, and the SUB bit and uses them for the 8-Bit RCA as C_{in}, A_{7-0}, $B_{eff_7-eff_0}$. The RCA itself will produce 2 outputs, S_{7-0}, C_{out}, the result (Sum) and the overflow (Carry-out). It is important to note that the range of the calculator is only 8 bits, meaning that it can only operate on values of $\{0, 255\}$. This means that if the user inputs values over that threshold it will cause the system to overflow. Subsequently, subtraction could only hold values of $\{-128, 127\}$ and values less than that threshold will also cause the system to overflow. Finally, the outputs will get processed once more by the BCD Decoder to 7-Segment. The outputs of the RCA will be processed and correctly added to the 7-Segment if the overflow value wasn't activated.

Finally, the output stage receives the output of the BCD Decoder and correctly displays the final result, human-readable decimal.

It is also crucial to note that the Minecraft version that was used is 1.18.2.

III.B. 7-Segment BCD, BCD Decoder

First, we note the truth table for each input from 0-15 and its outputs, a, b, c, d, e, f, g.

Decimal	Input				Output						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

TABLE VII

TRUTH TABLE FOR BCD TO 7-SEGMENT DECODER

Then, we simplify the K-Maps of each of the outputs,

		CD			
		00	01	11	10
AB	00	1	0	1	1
	01	0	1	1	1
	11	X	X	X	X
	10	1	1	X	X

TABLE VIII

K-MAP FOR SEGMENT a

		CD			
		00	01	11	10
AB	00	1	1	1	1
	01	1	0	1	0
	11	X	X	X	X
	10	1	1	X	X

TABLE IX

K-MAP FOR SEGMENT b

		CD			
		00	01	11	10
AB	00	1	1	1	0
	01	1	1	1	1
	11	X	X	X	X
	10	1	1	X	X

TABLE X

K-MAP FOR SEGMENT c

		CD			
		00	01	11	10
AB	00	1	0	1	1
	01	0	1	0	1
	11	X	X	X	X
	10	1	1	X	X

TABLE XI

K-MAP FOR SEGMENT d

		CD			
		00	01	11	10
AB	00	1	0	0	1
	01	0	0	0	1
	11	X	X	X	X
	10	1	0	X	X

TABLE XII

K-MAP FOR SEGMENT e

		CD			
		00	01	11	10
AB	00	1	0	0	0
	01	1	1	0	1
	11	X	X	X	X
	10	1	1	X	X

TABLE XIII

K-MAP FOR SEGMENT f

		CD			
		00	01	11	10
AB	00	0	0	1	1
	01	1	1	0	1
	11	X	X	X	X
	10	1	1	X	X

TABLE XIV

K-MAP FOR SEGMENT g

Results in these boolean functions,

$$a = A + C + BD + B'D'$$

$$b = B' + CD + C'D'$$

$$c = B + D + C'$$

$$d = A + B'C + B'D' + CD' + BC'D$$

$$e = B'D' + CD'$$

$$f = A + BC' + BD' + C'D'$$

$$g = A + BC' + B'C + CD'$$

To map these results in Minecraft, we would need to use a BCD Decoder that maps each of the outputs from their binary.

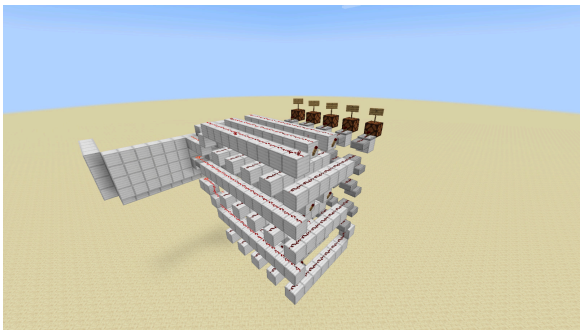


Fig. 8. Redstone implementation of the BCD to 7-segment decoder in Minecraft 1.18.2, showing gate networks for each segment output.

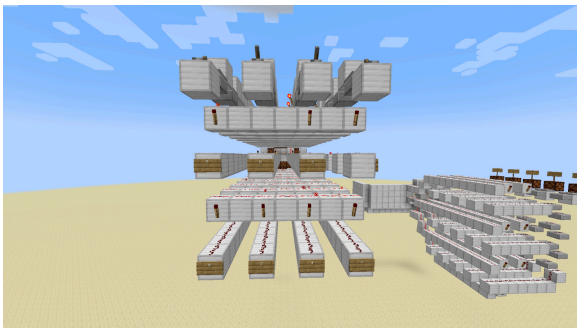


Fig. 8. Redstone implementation of the BCD to 7-segment decoder in Minecraft 1.18.2, showing gate networks for each segment output.

Fig. 8. Redstone implementation of the BCD to 7-segment decoder in Minecraft 1.18.2, showing gate networks for each segment output.

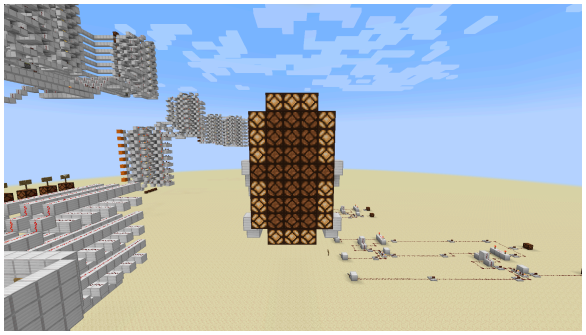


Fig. 8. Redstone implementation of the BCD to 7-segment decoder in Minecraft 1.18.2, showing gate networks for each segment output.

III.C. Half Adder, Full Adder, CCA Adder

The processing stage starts from here, from the basics of half adder, full adder, and the completed structure, Carry Cancel Adder (CCA). Half adder and full adder follow the same logical components that have been mentioned in section II.C, meanwhile the CCA Adder is similarly structured to a Ripple Carry Adder, only difference is that CCA is just a redstone optimization trick used for a RCA.

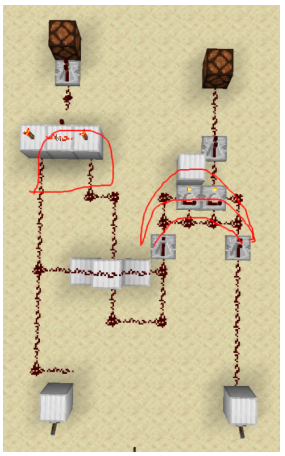


Fig. 12. Half adder implementation in Minecraft 1.18.2. XOR gate (top) produces the Sum output; AND gate (bottom) produces the Carry output.

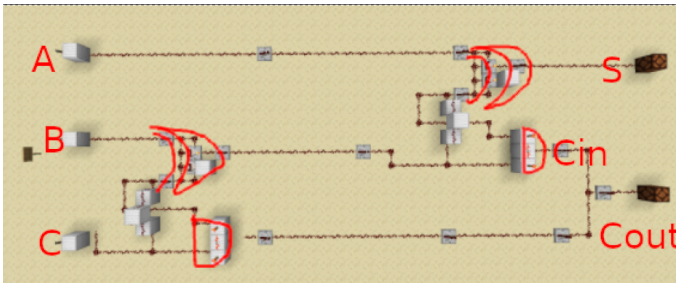


Fig. 13. Full adder implementation in Minecraft 1.18.2, composed of two half adder stages and one OR gate combining carry outputs C_1 and C_2.

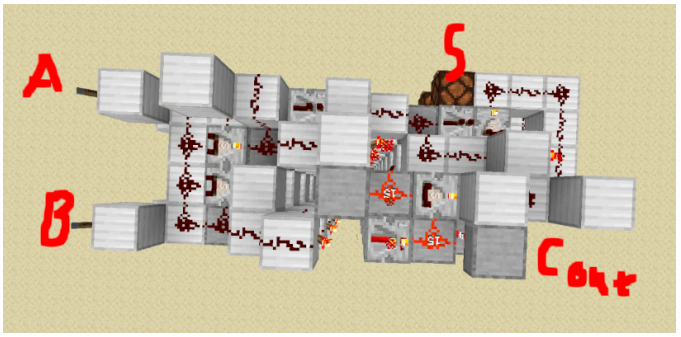


Fig. 14. 8-bit Carry Cancel Adder (CCA) implementation in Minecraft 1.18.2. Each stage processes one bit with carry signals propagating vertically across the chain.

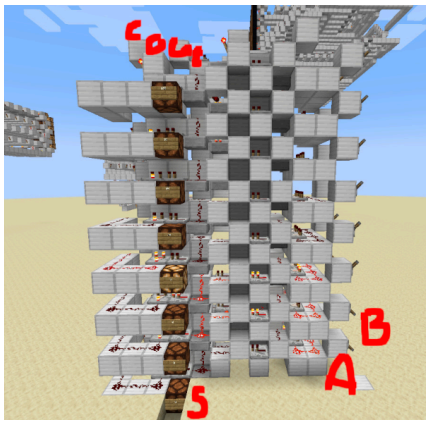


Fig. 15. 8-bit Carry Cancel Adder (CCA) implementation in Minecraft 1.18.2. Each stage processes one bit with carry signals propagating vertically across the chain.

III.D. Two's Complement

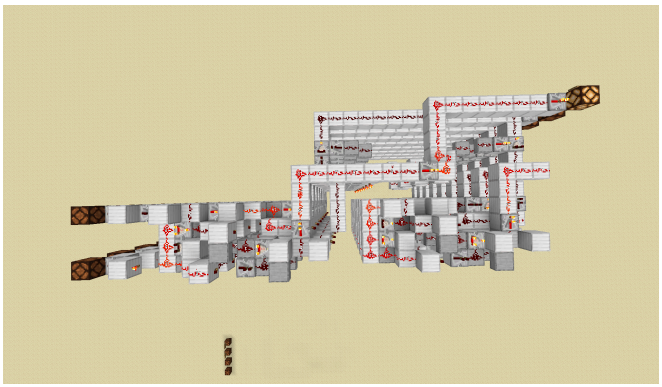


Fig. 16. Two's complement subtraction unit in Minecraft 1.18.2. The SUB control signal feeds into XOR gates on the B input line and sets Cin = 1 on the least significant adder stage, enabling subtraction via two's complement negation.

The two's complement subtraction unit extends the 8-bit ripple carry adder described in Section III.C without requiring additional adder hardware. The adder therefore computes A + (¬B) + 1, which is arithmetically equivalent to A − B by the definition of two's complement representation. This design eliminates the need for a dedicated subtraction circuit, demonstrating that a single Boolean control signal is sufficient to reconfigure the same combinational logic for two distinct arithmetic operations. The circuit was verified by testing the operation 7 − 3, yielding the binary result 0100 (decimal 4), consistent with the expected output derived analytically in Section II.C.

IV. Conclusion

This paper has presented the core design and implementation of a 8-bit binary calculator made within the Minecraft Redstone environment. This paper also utilizes the concepts of Boolean algebra, Binary arithmetic, and karnaugh map minimization. The calculator was built from core concepts such as half adders, full adders, and ripple carry adders, capable of doing operations such as addition and subtraction on 2 8-bit inputs.

The implemented calculator successfully produced results for all tested addition and subtraction cases. The Two's Complement subtraction mechanism demonstrated that a single arithmetic circuit, which is controlled by the SUB bit, is sufficient to perform both addition and subtraction without additional hardware. The BCD to 7-Segment Decoder, derived from karnaugh maps minimization of Boolean functions, correctly displayed output for all valid digits.

The implementation demonstrates that abstract concepts in discrete mathematics have direct, tangible realizations. Boolean algebra, which provides the theoretical foundation for digital logic, was shown to correspond directly to physical Redstone gate configurations. The Karnaugh map minimization process proved practically significant, reducing the number of logic gates required for the 7-segment decoder and simplifying the final redstone contraption.

Several limitations appeared while making this current design. The calculator is constrained to 8-bits, this makes it so that the range of values could only be produced to $\{0, 511\}$. Operations such as multiplication and division are also outside of the scope of this system. Future work could extend the design to support more bits and implement multiplication and division into the system.

Github Page

https://github.com/TSZCodes/8-bit-calculator-minecraft

Acknowledgment

I would like to formally thank those who helped me finish this paper. A special thanks to:

1. Allah SWT and Prophet Muhammad PBUH.
2. Both of the writer's parents.
3. Lecturer of IF1220 Discrete Mathematics, Rinaldi Munir
4. Writer's friends
5. SSKN
6. Vertera Family (STEI-K'25)

References

[1] GeeksforGeeks, "Arithmetic operations of binary numbers," GeeksforGeeks, Apr. 14, 2020. [Online]. Available: https://www.geeksforgeeks.org/digital-logic/arithmetic-operations-of-binary-numbers/. [Accessed: Jun. 19, 2026].

[2] mattbatwings, "Redstone Calculator Tutorial" YouTube, 2021. [Online]. Available: https://www.youtube.com/watch?v=lp54Dwasg5k&list=PL5LiOvrbVo8kDJ-wJu7l-lIc97yaZo96n. [Accessed: Jun. 19, 2026].

[3] GeeksforGeeks, "BCD to 7 segment decoder," GeeksforGeeks, Jan. 19, 2018. [Online]. Available: https://www.geeksforgeeks.org/digital-logic/bcd-to-7-segment-decoder/. [Accessed: Jun. 19, 2026].

[4] D. Llamocca, M. Garcia, J. Lee, and T. Schmidt, "BCD to binary converter," Oakland University, Dept. of Electrical and Computer Engineering, Tech. Rep., 2019. [Online]. Available: https://www.secs.oakland.edu/~llamocca/Courses/ECE2700/F19/FinalProject/Group7_bcd2bin.pdf. [Accessed: Jun. 19, 2026].

[5] mattbatwings, "CCA adder tutorial - the ultimate redstone adder," YouTube, 2021. [Online]. Available: https://www.youtube.com/watch?v=s5xBDcK84cM. [Accessed: Jun. 19, 2026].

[6] GeeksforGeeks, "Number system in maths," GeeksforGeeks, Sep. 22, 2020. [Online]. Available: https://www.geeksforgeeks.org/maths/number-system-in-maths/. [Accessed: Jun. 19, 2026].

[7] GeeksforGeeks, "Seven segment displays," GeeksforGeeks, Apr. 13, 2020. [Online]. Available: https://www.geeksforgeeks.org/digital-logic/seven-segment-displays/. [Accessed: Jun. 19, 2026].

[8] GeeksforGeeks, "Implementation of full adder using half adders," GeeksforGeeks, Jul. 23, 2025. [Online]. Available: https://www.geeksforgeeks.org/digital-logic/implementation-of-full-adder-using-half-adders/. [Accessed: Jun. 19, 2026].

[9] A. Arham, "Rancangan dan implementasi dekoder biner (BCD) ke 7-segmen menggunakan aljabar Boolean pada permainan Minecraft," undergraduate paper, Institut Teknologi Bandung, Bandung, Indonesia, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025-2/Makalah2025/Makalah-Matdis-2025-IF-ITB%20(101).pdf. [Accessed: Jun. 19, 2026].

[10] L. C. Kesuma, "Analisis end-to-end encryption dengan Diffie-Hellman key exchange menggunakan prinsip aljabar Boolean dan teori bilangan," undergraduate paper, Institut Teknologi Bandung, Bandung, Indonesia, 2022. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2022-2023/Makalah2022/Makalah-Matdis-2022%20(5).pdf. [Accessed: Jun. 19, 2026].

[11] CircuitVerse, "4-bit ripple carry adder," CircuitVerse, 2021. [Online]. Available: https://circuitverse.org/users/290531/projects/4-bit-ripple-carry-adder-60bd591e-2cea-453d-8016-2c24eeacd4da. [Accessed: Jun. 19, 2026].

[12] mattbatwings, "8-bit redstone calculator," Planet Minecraft, Apr. 5, 2020. [Online]. Available: https://www.planetminecraft.com/project/8-bit-redstone-calculator-4537773/. [Accessed: Jun. 19, 2026].

[13] R. Munir, "Aljabar Boolean bagian 2," Institut Teknologi Bandung, Bandung, Indonesia, lecture slides, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/11-Aljabar-Boolean-(2026)-bagian2.pdf. [Accessed: Jun. 19, 2026].

[14] M. Sharma, "Digital logic gate ICs with symbols and truth tables," *BragitOff.com*, Oct. 2015. [Online]. Available: https://www.bragitoff.com/2015/10/digital-logic-ics-with-symbols-and-truth-tables/. [Accessed: Jun. 19, 2026].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2025

Rifqi Irfan Indrawan 13525061