



WILHELM-LÖHE-SCHULE

EVANGELISCHE KOOPERATIVE GESAMTSCHULE

GRUNDSCHULE · MITTELSCHULE · REALSCHULE · FACHOBERSCHULE · GYMNASIUM

Oberstufenjahrgang 2024-2026

Abiturjahrgang 2026

Seminararbeit W-Seminar

Rahmenthema des W-Seminars: Ist doch logisch oder? Logik in der Mathematik und
Rhetorik

Thema der Seminararbeit: Von Logikgattern zu Speicherstrukturen:
Logikoperatoren in der Elektronik

Verfasser/in: Noah Schuller

Kursleiter/in: Oliver Mandel

Abgabetermin für Schüler: Dienstag, der 11.11.2025

Bewertung	Note	Notenstufe in Worten	Notenpunkte		Punkte
Schriftliche Arbeit				x 3	
Abschlusspräsentation mit Prüfungsgespräch				x 1	
Summe:					
Gesamtleistung nach § 29 Abs. 6 GSO = Summe : 2 (gerundet)					

Datum und Unterschrift der Kursleitung

Von Logikgattern zu Speicherstrukturen: Logikoperatoren in der Elektronik

Noah Schuller

8. November 2025

Inhaltsverzeichnis

1 Einleitung	4
2 Formallogische Grundlagen	4
2.1 Logikoperatoren	4
2.2 Wahrheitstabellen	6
3 Logikgatter und deren technische Implementierung	7
3.1 Transistoren	7
3.2 Beispiel: NOR-Gatter	8
4 Konstruktion elementarer logischer Schaltungen	8
4.1 Flip-Flops	9
4.1.1 S-R-Flip-Flops	9
4.1.2 J-K-, T-, und D- Flip-Flops	10
4.2 Binärzähler	12
4.3 Encoder und Decoder	13
4.4 Multiplexer und Demultiplexer	14
4.5 Schaltkreise zur Zeitverzögerung	15
5 Minecraft als Modell für logische Schaltungen	15
5.1 Einführung in Redstone-Mechaniken	16
5.2 Umsetzung der Logikgatter in Minecraft	18
5.3 Einschränkungen des Modells	18
6 Konstruktion von logischen Schaltungen in CircuitVerse	19

7 Umsetzung des Tabellengenerator-Modells	19
7.1 Generieren einer Tabellenzeile	19
7.2 Koordination der Abläufe	22
7.3 Zentrale Steuerung	23
7.4 Benutzerschnittstelle	24
8 Test und Evaluation	24
8.1 Test der beiden Modelle	24
8.2 Ergebnisse und Auswertung	25
8.3 Diskussion: Einordnung in die Von-Neumann-Architektur	28
9 Fazit	29
A Anmerkungen zur Konstruktion logischer Schaltungen in den beiden Modellen	30
A.1 Vereinfachungen durch Minecraft-Mechaniken	30
A.2 Sub-Circuits in CircuitVerse	33
B Unterschiede in der Umsetzung zwischen den beiden Modellen	36
B.1 Generieren einer Tabellenzeile	36
B.2 Koordination der Abläufe	37
B.3 Zentrale Steuerung	37
B.4 Benutzerschnittstelle	38
C Ergänzendes Material	42
D Anlagen	44
D.1 Dokumentation der Nutzung von KI	44
D.2 Download-Links	45

1 Einleitung

Den Innovationen der Digitaltechnik sind auch heute noch keine Grenzen gesetzt. So verdoppeln sich die Zahl der Transistoren in Chips und damit auch deren Rechenleistung und Speicherkapazität schon seit Jahrzehnten in konstanten Abständen von etwa zwei Jahren (Hellmann, 2022, 5 S. 6). Auch wenn diese Zahl heutzutage bereits im Milliardenbereich liegt, basieren selbst die komplexesten Supercomputer auf denselben Grundlagen.

Ziel dieser Arbeit soll sein, mithilfe der elementaren Operatoren der Logik nachzuvollziehen, wie sämtliche Schaltkreise eines Computers auf einfachen Logikfunktionen basieren. Aus diesem Grund werden nach einer kurzen Einführung in die logischen Grundlagen Schritt für Schritt komplexere Schaltkreise konstruiert, um schließlich ein System zu entwickeln, das sich elementarer Prinzipien der Computerarchitektur bedient. Zu diesem Zweck wird ein Wahrheitstabellengenerator entwickelt, der aus der Eingabe verknüpfter logischer Funktionen die entsprechende Wahrheitstabelle erzeugt. Um die Konstruktion desselben zu erleichtern und seine Funktionsweise zu prüfen, wird dieser modellhaft in zwei verschiedenen Systemen konstruiert. Die Entwicklung 10 des Generators wurde zunächst in Minecraft umgesetzt, da logische Schaltungen ähnlich zur physischen Hardware-Entwicklung von Grund auf neu entwickelt werden müssen. Zum Testen der Übertragbarkeit auf praxisnähere Umgebungen wurde das System danach ebenfalls in CircuitVerse, einem Online-Schaltkreis-Simulator, nachgebaut. All dies soll dem Zweck dienen, die Funktionsweise moderner Digitaltechnik anschaulich darzustellen und dem Leser so einen Einblick 15 in die Rechnerarchitektur zu gewähren. Außerdem soll die Frage geklärt werden, welche grundlegenden Elemente nötig sind, um die Konstruktion von Rechnern zu ermöglichen.

2 Formallogische Grundlagen

2.1 Logikoperatoren

Ein Logikoperator ist eine Funktion, die aus mindestens einer Aussage einen Wahrheitswert 25 liefert. In der Elektrotechnik werden für logische Aussagen auch die Begriffe „Schaltvariablen“ oder einfach „Signale“ synonym verwendet, während Logikoperatoren durch Schaltfunktionen bzw. Logikgatter umgesetzt werden. Eine Schaltfunktion eines Computers kann nach Eingabe der Eingangssignale den Wahrheitswert der Funktion ausgeben. Hierbei entspricht 0 „falsch“ bzw. „low“ und 1 „wahr“ bzw. „high“ (Reichardt, 2009, S.22).

Symbol	Wortbedeutung (Buchstabenkürzel)	Verhalten
$\neg A$	Nicht / NOT / Negation (n)	Invertiert die Aussage („Wahr“ wird zu „Falsch“ und umgekehrt)
$A \wedge B$	Und / AND / Konjunktion (u)	Gibt „Wahr“ aus, wenn sowohl A als auch B wahr ist
$A \vee B$	Oder / OR / Disjunktion (o)	Gibt „Wahr“ aus, wenn mindestens eine der beiden Aussagen wahr ist
$A \oplus B$	Entweder-Oder / XOR (x)	$(A \vee B) \wedge \neg(A \wedge B)$
$A \rightarrow B$	Wenn...dann / Implikation (k)	$\neg(A \wedge \neg B)$
$A \leftrightarrow B$	Genau dann, wenn... / XNOR / Bikonditional (b)	$\neg(A \oplus B)$

Tabelle 1: Logische Operatoren und ihre Bedeutung

In Tabelle 1 sind die Logikoperatoren genannt, die mit Hilfe des Wahrheitstabellengenerators berechnet werden sollen. Außerdem sind die Buchstabenkürzel angegeben, mit denen die Operatoren später in das System eingegeben werden können. Eine Übersicht über die Schaltsymbole für gängige Logikgatter ist der Abbildung 2.1.0.1 zu entnehmen.

In der Computertechnik spielen vor allem die Negationen von AND und OR eine große Rolle. Diese NAND- bzw. NOR-Funktionen werden in der Hardwareproduktion bevorzugt eingesetzt und bieten den Vorteil, dass sie **funktional vollständig** sind (Reichardt, 2009, S. 31). Das bedeutet, dass jedes andere Logikgatter aus ihnen hergestellt werden kann.

Grund dafür sind die De Morganschen Gesetze, denn diese „beschreiben, wie eine UND- bzw. eine ODER-Verknüpfung umgeformt werden kann, wenn der ganze Ausdruck invertiert wird.“ (Reichardt, 2009, S.28).

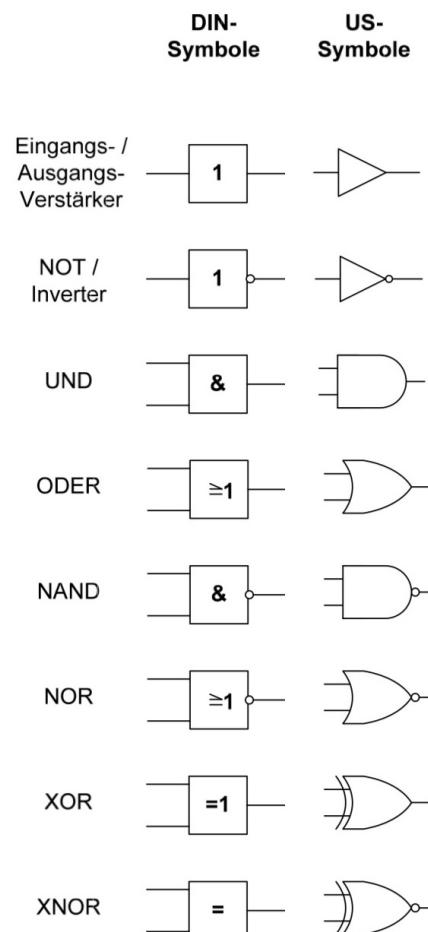


Abbildung 2.1.0.1: Logiksymbole der Boole'schen Algebra. © 2009 Oldenbourg Wissenschaftsverlag GmbH

Die De Morganschen Gesetze lauten:

$$\neg(A \wedge B) = \neg A \vee \neg B$$

$$\neg(A \vee B) = \neg A \wedge \neg B$$

So lässt sich beispielsweise der Operator \wedge nur durch NAND-Gatter umsetzen, indem ausgenutzt
⁵⁰ wird, dass $\neg x = \neg(x \wedge x) = x \text{ NAND } x$ gilt:

$$A \wedge B = \neg(A \text{ NAND } B) = (A \text{ NAND } B) \text{ NAND } (A \text{ NAND } B)$$

2.2 Wahrheitstabellen

Im Kontext der Aussagenlogik ist es oft nötig, eine Aussage auf deren Wahrheitsgehalt zu prüfen. Eine solche Aussage besteht meist aus mehreren elementaren Teilaussagen, die durch Logikoperatoren verknüpft sind. Dazu bedient man sich sowohl in der Formal- als auch in der Computerlogik sogenannter Wahrheitstabellen.
⁵⁵

„In Wahrheitstabellen werden auf der linken Seite alle möglichen Wertekombinationen der Eingangssignale aufgelistet. Auf der rechten Seite findet sich der Boole'sche Wert der Schaltfunktion. Dabei werden die Zeilen der Wahrheitstabelle so angeordnet, dass sie aufsteigenden Binärzahlen¹ [...] entsprechen“ (Reichardt, 2009, S.22).

⁶⁰ Als Beispiel sollen folgende Elementaraussagen betrachtet werden:

A: Es regnet draußen.

B: Ich muss das Haus verlassen.

C: Ich nehme einen Regenschirm mit.

Durch eine beliebige Verknüpfung der Operatoren ergibt sich folgende Aussage:

⁶⁵ $(A \wedge B) \rightarrow C \hat{=} \text{„Wenn es draußen regnet und ich das Haus verlassen muss, dann nehme ich einen Regenschirm mit.“}$

Die dritte Zeile der Tabelle 2 enthält beispielsweise die Information, dass, wenn A und C falsch sind und B wahr ist, $A \wedge B$ falsch ist, wodurch die gesamte Aussage korrekt ist.

Mithilfe einer Wahrheitstabelle kann man einerseits den Wahrheitsgehalt einer Gesamtaussage
⁷⁰ in Abhängigkeit der Werte der Teilaussagen analysieren. Ist die Gesamtaussage immer falsch, handelt es sich um einen Widerspruch. Eine Aussage, die unabhängig der eingegebenen Variablen

¹Grundlegende Kenntnisse über das Binärsystem werden vorausgesetzt und können bei Bedarf unter Chip, 2025 nachgeschlagen werden.

A	B	C	$A \wedge B$	$(A \wedge B) \rightarrow C$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

Tabelle 2: Wahrheitstabelle für die Aussage $(A \wedge B) \rightarrow C$

immer richtig ist, nennt man Tautologie (Beckermann, A., 2014, S.80). Ein Beispiel dafür ist die Aussage $A \vee \neg A$.

- Analog dazu lässt sich durch das Betrachten aller möglichen Werte der Ein- und Ausgänge das Schaltverhalten eines Gatters überprüfen (Hellmann, 2022, S. 56). Umgekehrt ist es möglich, mithilfe einer Wahrheitstabelle ein Schaltnetz zu konstruieren, das das gewünschte Verhalten umsetzt².

3 Logikgatter und deren technische Implementierung

3.1 Transistoren

- Transistoren sind elektrische Bauteile, die aus Halbleitern bestehen. Dabei wird vor allem zwischen Feldeffekttransistoren (FET) und Bipolartransistoren (BJT) unterschieden. BJTs haben je einen Kollektor, über den der Hauptstrom zum Emitter fließt, und eine Basis³ (vgl. Abbildung 3.1.0.1). Für einen npn-Transistor, eine Art von Bipolartransistor, gilt, dass der Kollektorstrom I_C nur dann fließt, wenn auch ein Basisstrom I_B fließt. Der Basisstrom I_B fließt erst dann, wenn die Schwellspannung U_{BE} erreicht ist (Elektronik-Kompendium, o. D. a). Da Schwellspannung und Basisstrom deutlich kleiner als der Kollektorstrom sind, eignet sich das Bauteil besonders gut zum Schalten und Verstärken von Signalen.

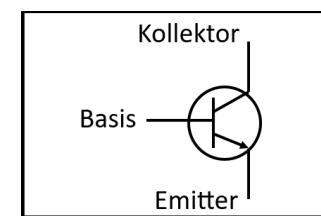


Abbildung 3.1.0.1: Schalt-Symbol eines Transistors. Eigene Darstellung.

²Auf eine detaillierte Darstellung der Konstruktionsmethodik wird an dieser Stelle zur Wahrung des thematischen Rahmens verzichtet.

³Für FETs werden analog die Begriffe Drain, Source und Gate verwendet (Reichardt, 2009, S. 144).

In modernen Computerschaltungen sind jedoch vorwiegend FETs verbaut (Reichardt, 2009, S.143). Die kombinierte Verwendung von NMOS- und PMOS-Transistoren, einer Unterkategorie der Feldeffekttransistoren, wird als CMOS-Technologie bezeichnet. Diese ist sehr energieeffizient, 95 denn „CMOS-Schaltungen haben keinen Stromverbrauch, solange an keinem Ausgang Strom fließt und keine Taktfrequenz ein CMOS-Teil schaltet“ (Elektronik-Kompendium, 2017). Ein NMOS-Transistor verhält sich ähnlich wie ein npn-Transistor, jedoch lässt er den Strom nicht bei einem fließenden Basisstrom durch, sondern wenn am Eingang eine positive Spannung anliegt (Reichardt, 2009, S.144). Mithilfe von Transistoren lassen sich also Logikgatter konstruieren, die 100 wiederum zum Prüfen von logischen Aussagen nötig sind.

3.2 Beispiel: NOR-Gatter

Durch den stark vereinfachten Schaltplan 3.2.0.1 wird das Konzept verdeutlicht, wie aus Transistoren ein Logikgatter entsteht. Sind beide Eingänge (A, B) inaktiv, blockieren die Transistoren den Stromfluss. Die positive Spannungsquelle verbindet sich direkt mit dem Ausgang X, sodass 105 dieser aktiv ist. Wird mindestens einer der beiden Eingänge aktiviert, fließt der Strom direkt zur Masse ab, wodurch das Schaltverhalten eines NOR-Gatters umgesetzt wird (theoryCIRCUIT, 2024).

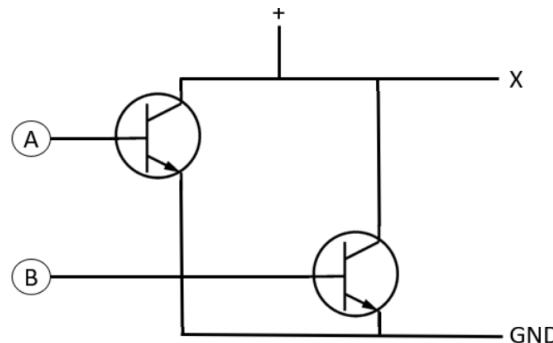


Abbildung 3.2.0.1: Vereinfachte NOR-Schaltung mithilfe von BJTs. Eigene Darstellung.

4 Konstruktion elementarer logischer Schaltungen

Um die Funktionsweise der einzelnen Bauteile, die für den Wahrheitstabellengenerator verwendet werden, zu erläutern, wird im Folgenden die vereinfachte Funktionsweise einiger wichtiger Schaltungen näher beschrieben. Da es teilweise mehrere Möglichkeiten gibt, das erwünschte Schaltverhalten umzusetzen, wird hier nur eine dieser Varianten genannt. Dadurch kann es zu Abweichungen zur realen Umsetzung mancher Bauteile kommen, jedoch ist eine modellhafte Betrachtung im Rahmen dieser Arbeit völlig ausreichend. 110

¹¹⁵ **4.1 Flip-Flops**

Ein Anwendungsbereich für Logikgatter sind Flip-Flops, welche die Grundlage fast aller digitalen Speichersysteme bilden. Allgemein ist ein Flip-Flop „[j]ede elektronische Schaltung, die zwei stabile elektrische Zustände hat und durch entsprechende Eingangssignale von einem Zustand in einen anderen geschaltet werden kann“ (Elektronik-Kompendium, o. D. b). Jedes Flip-Flop hat ¹²⁰ zwei Ausgänge: Den Hauptausgang Q und dessen Negation \bar{Q} , die immer exakt den umgekehrten Wert besitzt wie Q .

Flip-Flops können sich hinsichtlich Funktionalität und Aufbau leicht unterscheiden, weshalb es diverse Unterkategorien dieser Schaltkreise gibt.

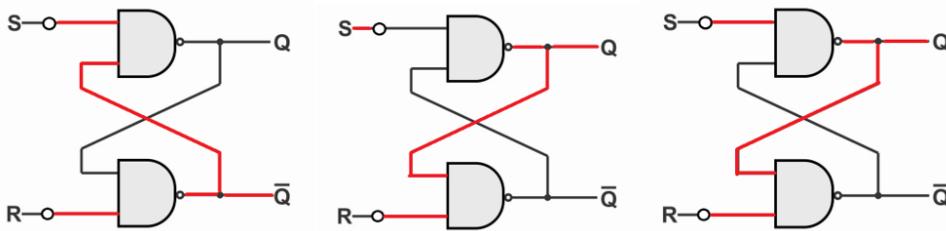
4.1.1 S-R-Flip-Flops

¹²⁵ Das Set-Reset-Flip-Flop (kurz: S-R-Flip-Flop) stellt die Grundlage für die weiteren Arten von Flip-Flops dar und ist das einfachste dieser Art von Speicherelementen. Die möglichen Zustände des Schaltkreises, die sich aus allen Kombinationen der zwei Eingänge S und R ergeben, sind in der Tabelle 3 zusammengefasst. Wie man erkennen kann, setzt der Set-Eingang den stabilen Zustand auf 1, während der Reset-Eingang diesen auf 0 zurücksetzt. Ist keiner der beiden Eingänge ¹³⁰ aktiv, so verändert sich auch am Ausgangswert nichts. Um dieses Schaltverhalten umzusetzen, benötigt man nur wenige grundlegende Logikgatter, wie man an der in Abbildung 4.1.1.1 dargestellten Schaltung erkennen kann.

Zur Veranschaulichung kann das Flip-Flop mit einem kippbaren Lichtschalter verglichen werden. Das Drücken der oberen Hälfte des Schalters entspricht der Aktivierung des S -Eingangs, das ¹³⁵ der unteren Hälfte der Aktivierung des R -Eingangs. Nach dem einmaligen Betätigen der oberen Hälfte bleibt der Schalter in dieser Position, bis er zurückgesetzt wird. Ähnlich wie beim Kippschalter gibt es keinen Mittelzustand, weshalb $S=R=1$ (gleichzeitiges Setzen und Zurücksetzen) nicht definiert ist. Das ist ein Nachteil dieser Schaltung, da eine solche Eingabe einen fehlerhaften Ausgangswert erzeugen würde und Folgefehler im System verursachen könnte.

Betriebsmodus	S	R	Q (neu)	\bar{Q} (neu)	Auswirkung auf Ausgang
Verboten	1	1	?	?	Nicht definiert (verbauter Zustand)
Setzen	1	0	1	0	Q wird auf 1 gesetzt
Rücksetzen	0	1	0	1	Q wird auf 0 zurückgesetzt
Speichern (Halten)	0	0	Q (alt)	\bar{Q} (alt)	Ausgang bleibt unverändert

Tabelle 3: Wahrheitstabelle des S-R-Flip-Flops



Links: Stationärer Zustand $S=R=0$; **Mitte:** Aktivierung des Ausgangs Q ; **Rechts:** Deaktivierung des Eingangs S .

Legende:

	Aktive Leitung		Inaktive Leitung
	Invertierung (NOT)		NAND-Gatter

Abbildung 4.1.1.1: Set-Vorgang eines vereinfacht dargestellten S-R-Flip-Flops, bearbeitet. © Engr. Shahzada Fahad

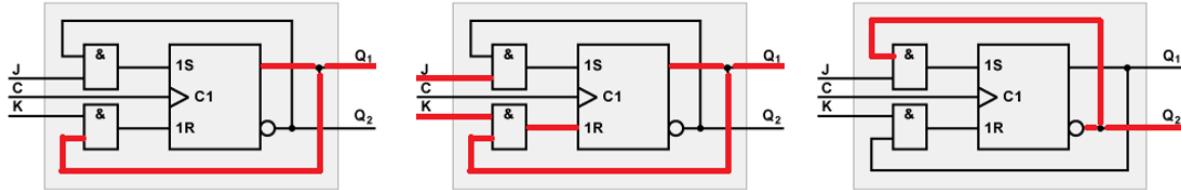
¹⁴⁰ **4.1.2 J-K-, T-, und D- Flip-Flops**

Das S-R-Flip-Flop lässt sich mit einem Kippschalter vergleichen. Analog hierzu ähnelt die Funktionsweise des T-Flip-Flops einem Druckschalter mit Rastfunktion, der bei jeder Betätigung das Licht abwechselnd an- und ausschaltet. Um diese Schaltfunktion durch Logikgatter umzusetzen, ist als Zwischenschritt die Konstruktion eines J-K-Flip-Flops nötig. Dieses stellt den Übergang ¹⁴⁵ vom S-R- zum T-Flip-Flop dar, da es beim Anlegen eines Taktimpulses seinen Ausgangszustand wechselt, wenn an beiden Eingängen (J und K) ein aktives Signal anliegt (Elektronik-Kompendium, o. D. c). Dadurch wird der nicht definierte Zustand $S=R=1$ eliminiert und es wird zusätzlich möglich, den Ausgangswert unabhängig vom aktuellen Zustand des Flip-Flops zu ändern. Umgesetzt wird dies durch die Erweiterung eines J-K-Flip-Flops um zwei zusätzliche ¹⁵⁰ AND-Gatter, die vom Ausgang zurück in die Eingänge speisen, wie in Abbildung 4.1.2.2 zu erkennen ist. Der Eingang C („Clock“) vereinfacht die Synchronisation, indem die Eingänge erst nach dem Taktimpuls eines zentral gesteuerten Clock-Signals verarbeitet werden. Außerhalb von größeren Computersystemen ist das jedoch zunächst nicht relevant.

Für das Toggle-Flip-Flop (kurz: T-Flip-Flop) muss der Schaltkreis des J-K-Flip-Flops erweitert ¹⁵⁵ werden. Dazu werden dessen Eingänge leitend verbunden, sodass sie gemeinsam den einzigen Eingang T der Schaltung darstellen. Das exakte Schaltverhalten ist in der Tabelle 4 abgebildet.

Betriebsmodus	T	Q (alt)	Q (neu)	\bar{Q} (neu)
Halten (Speichern)	0	0	0	1
Halten (Speichern)	0	1	1	0
Umschalten (Toggle)	1	0	1	0
Umschalten (Toggle)	1	1	0	1

Tabelle 4: Wahrheitstabelle des T-Flip-Flops



Links: Stationärer Zustand $J=K=0$; **Mitte:** Aktivieren beider Eingänge; **Rechts:** Umschalten (Toggle) des Ausgangs Q

Legende:

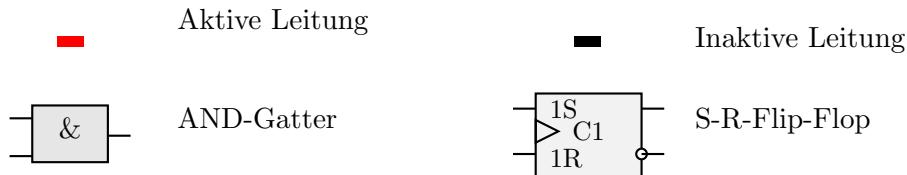


Abbildung 4.1.2.2: Toggle-Vorgang eines J-K-Flip-Flops, bearbeitet. © Elektronik-Kompendium

Auch ein D-Flip-Flop besteht in seinen Grundzügen aus einem S-R-Flip-Flop, jedoch wird hier der einzige Eingang E gleichzeitig in den Set-Eingang und invertiert in den Reset-Eingang gespeist. Dadurch wird der illegale Zustand des S-R-Flip-Flops ebenfalls eliminiert (vgl. Abbildung 4.1.2.1). Diese Art von Flip-Flop speichert bei jedem Signal der Clock den aktuellen Eingang in den Ausgang Q. Ist die Clock nicht aktiv, ändert sich auch am Ausgang nichts. Im Tabellengenerator kommen vorwiegend T- und S-R-

Flip-Flops vor, da die Reset- und die Toggle-Funktion eines Flip-Flops kaum an derselben Stelle benötigt werden. D-Flip-Flops waren nur für die Tastatureingabe in CircuitVerse nötig (vgl. Anhang Kapitel A.2). S-R-Flip-Flops sind vor allem nützlich, da sie sich gut als Speicher für Zwischenwerte eignen und nach jedem Be-

rechnungszyklus einfach zurückgesetzt werden können.

Jedoch muss bei ihrer Implementierung auf die Vermeidung des illegalen Zustandes geachtet werden. T-Flip-Flops spielen vor allem in Schaltkreisen wie dem Binärzähler eine Rolle.

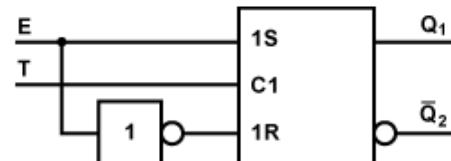


Abbildung 4.1.2.1: Vereinfachte Schaltung des D-Flip-Flops. © Elektronik-Kompendium

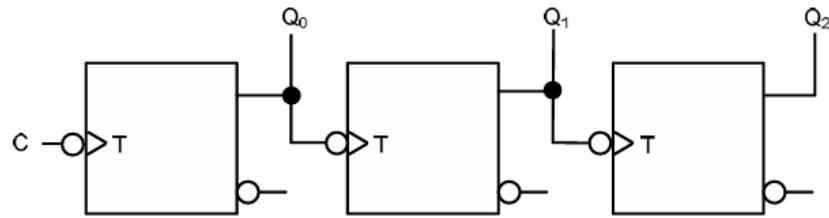


Abbildung 4.2.0.1: 3-Bit-Rückwärtszähler. © 2022 Walter de Gruyter GmbH

4.2 Binärzähler

Ein Binärzähler ist ein Schaltkreis, der einen Eingang und n Ausgänge besitzt, wobei ein Ausgang für je eine Stelle einer Binärzahl steht. Bei jedem Puls, den der Eingang erhält, wird die Binärzahl um 1 erhöht bzw. erniedrigt. Dies funktioniert durch eine Verkettung mehrerer T-Flip-Flops, deren Ausgänge jeweils den Eingang des nächsten schalten (Hellmann, 2022, S. 95). So speichert jeder Baustein je eine Stelle der Zahl. Verwendet man die Q -Ausgänge der Flip-Flops, zählt der Zähler rückwärts. Mit den invertierten Ausgängen $\neg Q$ ergibt sich ein vorwärtszählender Zähler. Interpretiert man die Ausgänge des Binärzählers aus Abbildung 4.2.0.1, so erhält man nach m Schaltvorgängen die in Tabelle 5 dargestellten Zahlen. Das Signal des Flip-Flops, das direkt mit dem Eingang verbunden ist, dient dabei als rechte sowie das n -te als linke äußerste Stelle der Binärzahl.

m	Rückwärtszähler		Vorwärtszähler	
	Binärzahl $Q_2Q_1Q_0$	Dezimaldarstellung	Binärzahl $Q_2Q_1Q_0$	Dezimaldarstellung
0	111	7	000	0
1	110	6	001	1
2	101	5	010	2
3	100	4	011	3
4	011	3	100	4
5	010	2	101	5
6	001	1	110	6
7	000	0	111	7

Bei $m = 8$ springen beide Zähler auf ihren jeweiligen Anfangswert zurück.

Tabelle 5: Ausgegebene Binärzahlen eines 3-Bit-Rückwärts- und Vorwärtszählers nach m Taktimpulsen.

Sind bei einem Vorwärtszähler alle Flip-Flops aktiviert, ist die ausgegebene Binärzahl 0, da der invertierte Ausgang $\neg Q$ verwendet wird. Wird das erste Flip-Flop zweimal aktiviert, ändert sich der Ausgang zunächst auf 1. Beim anschließenden Zurücksetzen auf 0 schaltet das Flip-Flop gleichzeitig die zweite Stelle, wodurch diese den Wert 1 annimmt. Dadurch wird jedes Mal,

wenn die vorherige Ziffer von 1 auf 0 springt, ein Signal an die nächste Stelle weitergeleitet. Dies
 190 realisiert den Übertrag beim Zählen. Nach $2^n - 1$ Schaltvorgängen ist die größtmögliche Zahl
 erreicht, was der Fall ist, wenn jeder Ausgang 1 anzeigt. Durch das nächste Schalten werden alle
 Flip-Flops gleichzeitig aktiviert, wodurch der ganze Binärzähler auf 0 zurückgesetzt wird.
 Binärzähler eignen sich nicht nur zum Zählen, sondern auch zum Darstellen aller möglichen
 Kombinationen von n binären Variablen⁴. Das liegt daran, dass der Binärzähler bis zu seinem
 195 Maximalwert jede mögliche Ziffernkombination durchläuft.

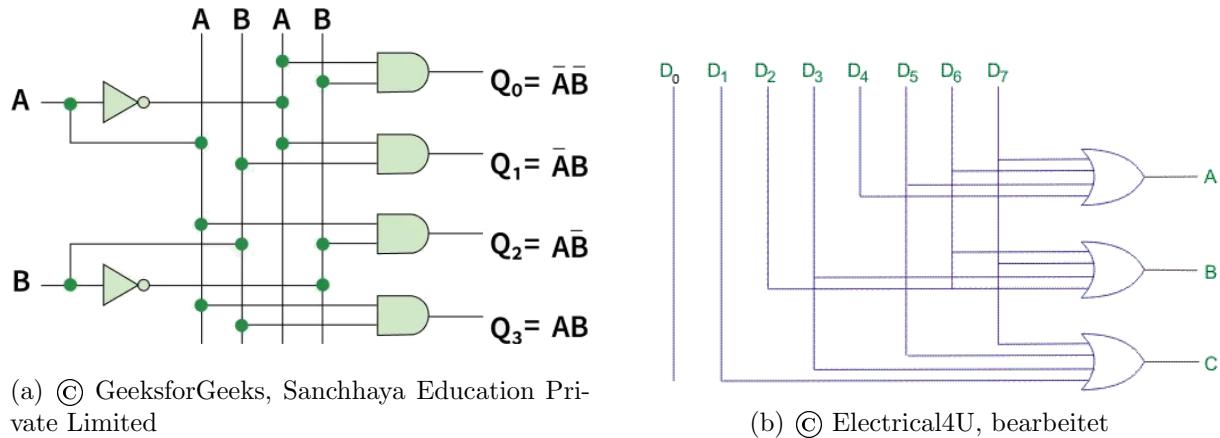
4.3 Encoder und Decoder

„Ein Binärzahlendecoder hat die Aufgabe durch Auswertung eines n -Bit-Auswahlsignals, einen
 von 2^n Ausgängen anzusteuern“ (Reichardt, 2009, S.170). Umgesetzt wird dies, indem vor jeden
 der Ausgänge des Decoders ein AND-Gatter geschaltet ist. Das liegt daran, dass ein Ausgang
 200 nur dann aktiviert werden soll, wenn jede einzelne Ziffer der Binärzahl mit der Nummer des
 Ausgangs übereinstimmt. Eine Stelle ist mit dem AND-Gatter des Ausgangs direkt verbunden,
 wenn deren Index als Binärzahl an dieser Stelle eine 1 hat. Ansonsten wird sie als Negation
 in das Gatter gespeist. Vor dem Ausgang 6 (Binär: 110) wird beispielsweise die rechte Stelle
 invertiert und der Rest auf übliche Weise in das AND-Gatter geleitet. Daraus ergibt sich die
 205 Schaltung 4.3.0.1a. Ein Decoder besitzt daher stets genau einen aktiven Ausgang. Das ist der
 Ausgang, dessen Index der eingegebenen Binärzahl entspricht.

Ein Encoder wandelt dieses Signal wieder in ein n -Bit-Signal um, indem er die aktivierte Leitung
 in eine entsprechende Binärzahl überführt. Dazu wird der Eingang x mithilfe von OR-Gattern in
 alle Ausgänge geleitet, die aktiv sein müssen, damit sich die Binärzahl x ergibt (vgl. Abbildung
 210 4.3.0.1b). Hierbei ist wichtig, dass immer höchstens ein Eingang aktiviert wird, da die Ausgabe
 sonst fehlerhaft wird.

Oftmals werden zur kompakten Übertragung Signale erst codiert und später wieder decodiert.
 Durch die binäre Codierung bleibt der Informationsgehalt gleich, während die Anzahl der Leitungen von 2^n auf n reduziert wird.
 215 Verbindet man die Ausgänge eines Binärzählers mit den Eingängen eines Decoders, ist genau der Ausgang des Decoders aktiv, der dem aktuellen Zählindex des Binärzählers entspricht. Dadurch kann ein Signal weitergegeben werden, wenn ein Binärzähler genau m -mal aktiviert wurde. Das ist der Fall, wenn der m -te Ausgang des Decoders ein aktives Signal liefert. Dies ist beispielsweise für die Koordination von m -mal wiederholten Schleifen nützlich.

⁴Diese Anwendung ist zur Auswahl der Startwerte für die Elementaraussagen nötig (vgl. Kapitel 7.1)



(a) © GeeksforGeeks, Sanchhaya Education Private Limited

(b) © Electrical4U, bearbeitet

Abbildung 4.3.0.1: Schaltung eines 2-Bit-Decoders (links) und 8:3-Encoders (rechts).

220 4.4 Multiplexer und Demultiplexer

Ein $n : 1$ Multiplexer ist ein Umschalter, der aus n Eingängen genau einen auswählt und dieses Signal an den einzigen Ausgang weiterleitet. Dazu hat er zusätzlich zu seinen Haupteingängen noch $\log_2(n)$ weitere Steuereingänge (Hellmann, 2022, S. 76). Diese werden als Binärzahl interpretiert und mithilfe eines Decoders in n Leitungen umgewandelt, von denen genau eine aktiv ist.

225 Jeder Ausgang des Decoders wird zusammen mit den Eingängen des Multiplexers durch ein eigenes AND-Gatter geleitet. Dadurch wird der Durchgang des Signals ermöglicht, das an der Stelle steht, die mit der Binärzahl übereinstimmt. Im Beispiel 4.4.0.1 wird durch Eingabe der Binärzahl 10 über den 2-Bit-Eingang S der aktive Eingang 2 an den Ausgang weitergeleitet. Dadurch lässt sich die Funktionsweise eines Multiplexers anschaulich der eines Radios vergleichen, denn ein Radioempfänger spielt nur den Sender ab, dessen Nummer über das Steuersignal ausgewählt wurde.

230 Ein $1 : n$ Demultiplexer arbeitet invers zum Multiplexer und leitet das Eingangssignal an genau einen der n Ausgänge weiter, der über eine Binärzahl am Steuereingang festgelegt wurde. Dazu wird der einzige Eingang in n AND-Gatter geleitet, deren Ausgänge je einen Gesamtausgang des Demultiplexers darstellen. Auch hier wird das Steuersignal wieder über einen Decoder interpretiert.

240 Im Wahrheitstabellengenerator eignen sich Multiplexer vor allem zur Auswahl aktuell benötigter Zwischenwerte, wie unter Kapitel 7.1 erläutert wird.

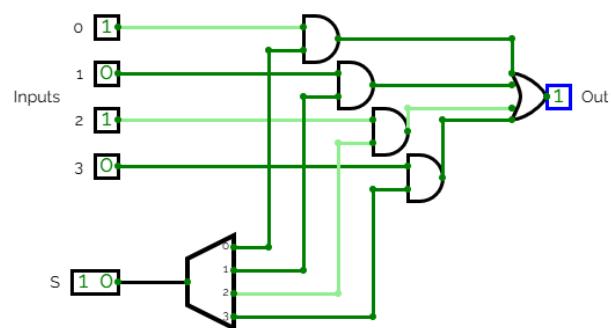


Abbildung 4.4.0.1: 4:1 Multiplexer. Screenshot einer eigens erstellten Schaltung; © CircuitVerse

4.5 Schaltkreise zur Zeitverzögerung

Bauteile, deren Schaltverhalten zeitabhängig ist, stellen insofern einen Sonderfall dar, dass sie im Gegensatz zu anderen bisher erläuterten Bauteilen in der Praxis nicht nur mit Logikgattern auskommen. Die Zeitverzögerung wird hier meist durch einen Widerstand R und einen Kondensator C implementiert, da sich letzterer vergleichsweise lange auf- oder entlädt, wohingegen Logikgatter eine sehr geringe Schaltverzögerung besitzen. Dadurch ist die Verzögerung besser vorhersagbar. In modernen Computersystemen ist, wie bereits in Kapitel 4.1.2 erwähnt, zeitliche Verzögerung über die Clock umsetzbar, die in regelmäßigen Intervallen Signale verschickt.

Ein Beispiel für zeitbasierte Schaltkreise sind sogenannte monostabile Kippstufen. „Eine monostabile Kippstufe ist eine Schaltung, die einen stabilen Zustand bzw. Ruhezustand besitzt. Dieser Zustand kann vorübergehend in einen anderen Zustand wechseln. Allerdings kehrt die monostabile Kippstufe nach einer bestimmten Zeit automatisch in den ursprünglichen stabilen Zustand zurück“ (Elektronik-Kompendium, o. D. d). In praktischen Hardwareimplementierungen werden diese durch ein integriertes R - C -Modul gesteuert. In den beiden Modellen für den Wahrheitstabellengenerator gibt es zwar keine solchen elektrischen Bauteile, jedoch können monostabile Kippstufen auf andere Art und Weise realisiert werden (vgl. Anhang Kapitel A.1 und A.2)

Im Wahrheitstabellengenerator ist eine präzise zeitliche Koordination essenziell, um fehlerhafte Zwischenergebnisse in verketteten oder wiederholt ausgeführten Prozessen zu vermeiden. Dazu muss berücksichtigt werden, dass jeder Prozess eine bestimmte Zeit zur Berechnung benötigt, bevor er das richtige Ergebnis liefern kann. Andererseits sorgen unverhältnismäßig lange Wartezeiten für eine insgesamt längere Verarbeitungsdauer, weshalb das Timing der Abläufe durch zahlreiche Tests abgestimmt werden muss.

5 Minecraft als Modell für logische Schaltungen

Minecraft ist ein blockbasiertes Sandbox-Computerspiel. Dies bedeutet, dass Spieler Blöcke, die vom Spiel bereitgestellt werden, beliebig kombinieren können (Minecraft-Wiki, 2025a). Die Konstruktion des Wahrheitstabellengenerators ist grundsätzlich auch ohne Vorkenntnisse über das Spiel nachvollziehbar. Zur Bedienung des Modells und zur genauen Betrachtung der einzelnen Schaltkreiselemente über die bereitgestellte Minecraft-Welt (siehe Anhang Kapitel D.2) sind jedoch zumindest grundlegende Kenntnisse über die Steuerung des Spielers nötig.

5.1 Einführung in Redstone-Mechaniken

In Minecraft lassen sich mit bestimmten Blöcken, die auf einem Rohstoff namens „Redstone“²⁷⁵ basieren, einfache elektrische Schaltkreise konstruieren.

Die **Redstone-Leitung** (☒⁵) funktioniert analog zu elektrischen Leitungen. Signale gehen von einer Quelle aus und werden mit zunehmendem Abstand vom Ursprung des Signals schwächer. Redstone-Leitungen lassen sich beispielsweise durch Hebel (☒) oder Knöpfe aktivieren, woraufhin sie optisch heller dargestellt werden (☒). Das Signal direkt an der Quelle hat eine maximale Stärke von 15. Mit jedem Block, den das Signal zurücklegt, vermindert sich diese Stärke um 1, bis sie wiederum 0 beträgt (vgl. Abbildung 5.1.0.1). Man kann mithilfe einer Redstone-Leitung ein OR-Gatter konstruieren, indem man zwei Eingänge über eine Leitung verbindet und den Ausgang dieser Leitung überwacht (vgl. Abbildung 5.1.0.3). Wird mindestens einer der Eingänge betätigt, wird der ganze Schaltkreis aktiviert.

²⁸⁰ Mithilfe eines **Redstone-Verstärkers** (☒) kann jede beliebige Signalstärke wieder auf 15 erhöht werden, wodurch theoretisch unbegrenzt lange Leitungen existieren können (vgl. Abbildung 5.1.0.1). Eine weitere Eigenschaft des Redstone-Verstärkers ist, dass er Signale verzögert weitergibt, jedoch mindestens mit einer festlegbaren Länge. Außerdem ist der Durchgang des Signals nur in einer Richtung möglich, wodurch er auch mit einer Diode vergleichbar ist (Minecraft-²⁹⁰ Wiki, 2025c).

Die **Redstone-Fackel** (☒) aktiviert standardmäßig benachbarte Redstone-Leitungen und kann so als Signalquelle dienen. Wird jedoch der Block, an dem diese befestigt ist, von einer aktiven Redstone-Leitung berührt, so erlischt sie und gibt kein Signal mehr weiter (Minecraft-Wiki, 2025b). Durch diese Funktion kann sie als NOT-Gatter verwendet werden (vgl. Abbildung ²⁹⁵ 5.1.0.1).

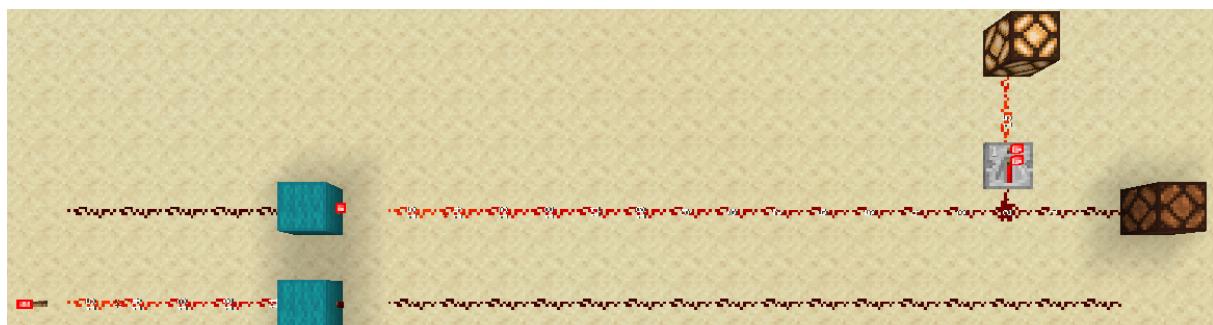


Abbildung 5.1.0.1: **Oben:** inaktive Leitung wird erst durch Fackel invertiert und nach 15 Blöcken verstärkt. **Unten:** aktive Leitung deaktiviert Redstone-Fackel.
Eigene Darstellung; © für Texturen: Mojang Studios

⁵© für Texturen von Minecraft-Blöcken: Mojang Studios

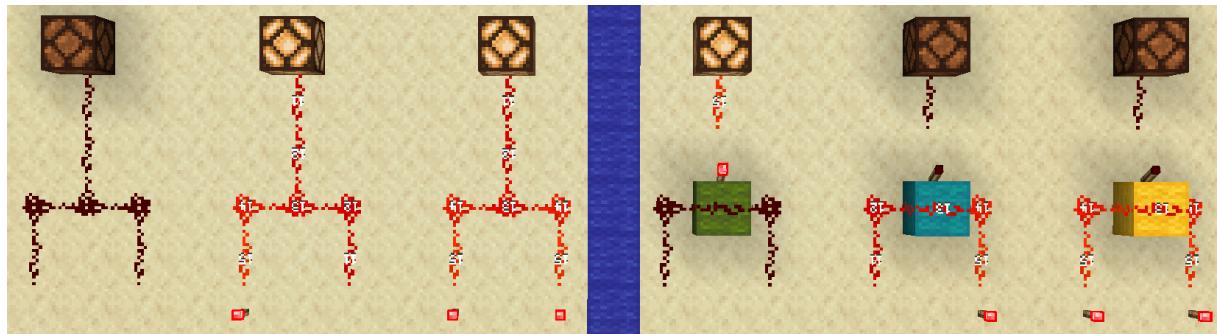


Abbildung 5.1.0.3: Drei OR-Gatter (links) bzw. NOR-Gatter (rechts), bei denen jeweils 0, 1 bzw. beide Eingänge aktiv sind. Eigene Darstellung; © für Texturen: Mojang Studios

Da Redstone-Fackeln in Minecraft im Gegensatz zu realen Logikgattern eine konstante, ausreichend große zeitliche Verzögerung besitzen, kann man mit diesen eine **Clock** bauen. Dazu werden Fackeln zyklisch verkettet, die sich gegenseitig periodisch aktivieren (vgl. Abbildung 5.1.0.2). Die Clock hat eine konstante Verzögerung in Redstone-Ticks. Ein Redstone-Tick ist eine festgeleerte Zeiteinheit von standardmäßig 100ms, mit denen viele Bauteile in Minecraft synchronisiert sind. Dadurch hat die Clock eine konstante Periodendauer von $3 * 2 * 1$ Redstone-Tick $\hat{=} 0,6s$. Das liegt daran, dass sich drei Redstone-Fackeln mit je einem Tick Verzögerung gegenseitig schalten. Der Faktor 2 kommt daher, dass die Fackeln nur bei jedem zweiten Zyklus von „Aus“ auf „An“ springen und dementsprechend nur jedes zweite Mal ein Clock-Signal ausgegeben wird. In sehr leistungsschwachen Umgebungen kann die Tickrate leicht schwanken, was minimale Abweichungen in der realen Periodendauer verursachen kann.

Verbindet man den Ausgang eines OR-Gatters mit einer Redstone-Fackel, so erhält man ein NOR-Gatter (vgl. Abbildung 5.1.0.3). Durch diese logische Verknüpfung ist bereits die **funktionalen Vollständigkeit** gewährleistet (vgl. Kapitel 2.1). Da außerdem eine verlässliche Clock nur mit Fackeln und Leitungen konstruiert werden kann, sind durch diese zwei Elemente bereits alle Bauteile aus Kapitel 4 umsetzbar. Ob diese ausreichen, ein computerähnliches System zu konstruieren, wird im Laufe der Arbeit noch diskutiert.

Im Tabellengenerator wurden dennoch einige zusätzliche Bauteile verwendet, die sich anderer Spielmechaniken bedienen. Diese implementieren jedoch keine neuen Funktionen, sondern ermöglichen lediglich eine kompakte Umsetzung von Schaltungen, die sonst deutlich aufwändiger wären. Die wichtigsten von diesen sind im Anhang unter Kapitel A.1 weiter ausgeführt.

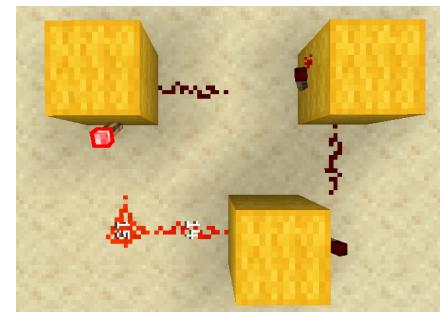


Abbildung 5.1.0.2: Clock basierend auf Redstone-Fackeln. Eigene Darstellung; © für Texturen: Mojang Studios

5.2 Umsetzung der Logikgatter in Minecraft

Da in Minecraft als grundlegende Logikgatter nur OR und NOT implementiert sind, müssen die übrigen Logikoperatoren z.B. durch die De Morganschen Gesetze in diese beiden Gatter umgeformt werden. Diese Umformungen sind in Tabelle 6 dargestellt. Die konkrete Umsetzung in Minecraft kann Abbildung C.0.0.1 des Anhangs entnommen werden.

Logikoperator	Umformung nur mit NOT und OR
\neg	$\neg A$
\wedge	$A \wedge B = \neg(\neg A \vee \neg B)$
\vee	$A \vee B$
\oplus	$A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B) \\ = \neg(\neg A \vee B) \vee \neg(A \vee \neg B)$
\rightarrow	$A \rightarrow B = \neg(A \wedge \neg B) = \neg A \vee B$
\leftrightarrow	$A \leftrightarrow B = (A \wedge B) \vee (\neg A \wedge \neg B) \\ = \neg(\neg A \vee \neg B) \vee \neg(A \vee B)$

Tabelle 6: Logikoperatoren und ihre Umformungen basierend auf den Operatoren NOT (\neg) und OR (\vee).

5.3 Einschränkungen des Modells

Auch wenn es durchaus möglich ist, vergleichsweise einfache Schaltungen in Minecraft nachzubauen, gibt es im Vergleich zu realer Rechnerarchitektur einige Einschränkungen, die beachtet werden müssen. Durch das Tick-System ist die größtmögliche Clock-Frequenz mit 10Hz äußerst limitiert. Im Gegensatz dazu liegt das Clock-Intervall von modernen PCs teilweise im GHz-Bereich und ist somit um mehrere Größenordnungen schneller.⁶

Außerdem weicht das System von Redstone wesentlich vom realen Konzept der Elektrizität ab. Während man in der realen Welt einen geschlossenen Stromkreis und eine konstante Energiezufuhr benötigt, breitet sich ein Redstone-Signal linear von einer Quelle aus und kann von dort beliebig oft verstärkt werden. Da ein Redstone-Verstärker mindestens 100ms Verzögerung besitzt, lassen sich Signale über weite Strecken außerdem nur vergleichsweise langsam übertragen. Schließlich sind die Systeme schwer skalierbar, da aufgrund von Rechnerleistung nur begrenzt viele Blöcke in einer Welt geladen sein können, wodurch die Anzahl von Transistoren in modernen Computerchips kaum realisierbar ist.

⁶Dies ist auch trotz maximaler Beschleunigung des Spiels auf ca. maximal 500 Redstone-Ticks pro Sekunde noch der Fall (vgl. Kapitel 8.2)

Um das System auch in einer realitätsnäheren Umgebung zu testen, die manche Nachteile der Umsetzung in Minecraft umgeht, soll der Wahrheitstabellengenerator ebenfalls in einer Schaltkreissimulationssoftware namens CircuitVerse repliziert werden.
345

6 Konstruktion von logischen Schaltungen in CircuitVerse

CircuitVerse ist ein digitaler Simulator, in dem logische Schaltkreise theoretisch beliebig groß konstruiert werden können (CircuitVerse, 2025b). Es sind hierfür mehrere Logikgatter schon als Bausteine vorhanden. Das gilt für alle Operatoren, die das System berechnen soll, mit Aufnahme des Konditional-Operators (\rightarrow). Dieser lässt sich jedoch durch andere Gatter konstruieren (vgl. Tabelle 1). Außerdem sind viele der logischen Schaltungen aus Kapitel 4 bereits als Bausteine vorgefertigt. Eine vollständige Dokumentation der bereitgestellten Elemente lässt sich unter CircuitVerse, 2025a finden. Es gibt auch die Möglichkeit, mehrere 1-Bit-Signale durch dieselbe Leitung zu übertragen, wodurch man z.B. mehrere 4-Bit-Signale durch einen Multiplexer auswählen kann. Die Anzahl der Bits pro Leitung wird als Bitbreite bezeichnet.
350
355

Durch sogenannte Sub-Circuits kann man oft benötigte Schaltkreiselemente einmal konstruieren und anschließend beliebig oft in ein größeres Projekt einbinden. Zur praktischen Implementierung des Wahrheitstabellengenerators war die Konstruktion mehrerer Sub-Circuits nötig. Deren Funktionsweise und Verkabelung können im Anhang unter Kapitel A.2 nachgeschlagen werden.

360

7 Umsetzung des Tabellengenerator-Modells

Abbildung C.0.0.2 des Anhangs enthält einen Überblick über das gesamte Tabellengeneratormodell in Minecraft sowie in CircuitVerse. Zusätzlich sind die Positionen wichtiger Bauteile farblich markiert. Auch wenn die Umsetzung in beiden Modellen sehr ähnlich ist, gibt es dennoch manche Unterschiede in der Umsetzung. Diese sind im Anhang unter Kapitel B ausgeführt.

365

7.1 Generieren einer Tabellenzeile

Der Wahrheitstabellengenerator ist darauf ausgelegt, bis zu acht Elementaraussagen (A-H) durch bis zu sieben Logikoperatoren (1-7) beliebig miteinander zu verknüpfen. Zunächst wird für die Aussagen A–H eine Kombination von Wahrheitswerten festgelegt, die als Grundlage für die Berechnung dient. Dazu wird der Binärzähler B verwendet, bei dem jede Stelle für einen solchen Startwert steht. So wird beispielsweise mit „wahr“ für die Aussage C gerechnet, wenn die dritte Stelle des Binärzählers 1 ist. Ist die Zeile vollständig berechnet, wird ein Signal an den Eingang des Binärzählers geschickt, wodurch dieser die nächste Kombination aufruft.
370

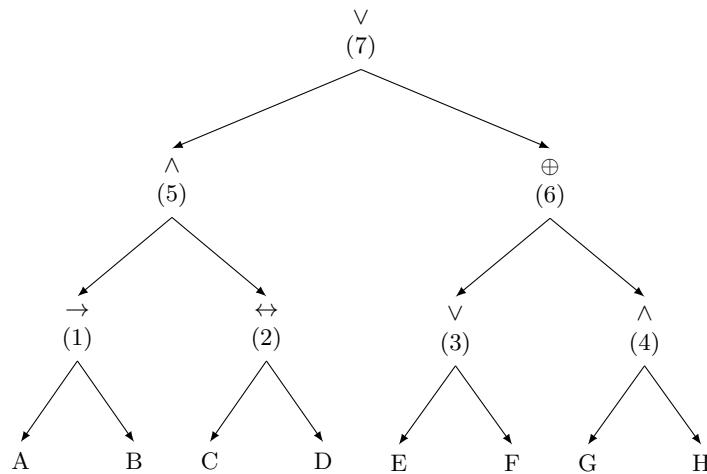


Abbildung 7.1.0.1:

Binärbaumdarstellung der Aussage X: $((A \rightarrow B) \wedge (C \leftrightarrow D)) \vee ((E \vee F) \oplus (G \wedge H))$. Die Zahl in Klammern stellt dabei die Reihenfolge der Berechnung dar. Eigene Darstellung.

Die Berechnung der Werte lässt sich modellhaft als Binärbaum darstellen. Ein Binärbaum ist eine Art von Graph, der bei dem sogenannten Wurzelknoten beginnt, der sich auf der ersten 375 Ebene befindet. Von jedem Knoten gehen maximal zwei Kanten aus, die wiederum mit Knoten auf der nächsten Ebene verbunden sind. Knoten, von denen keine weiteren Kanten ausgehen, werden auch Blätter genannt.

In diesem Fall ist der Wurzelknoten das letzte zu berechnende Logikgatter, während die Blätter 380 die Elementaraussagen sind. In der Umsetzung werden die acht Startwerte zu vier Paaren zusammengefasst, die jeweils durch ein Logikgatter verknüpft werden. Im nächsten Schritt werden die Wahrheitswerte der vier Operatoren berechnet. Dieser Prozess wird für die vier neu entstandenen Teilaussagen wiederholt, bis im letzten Schritt der Wahrheitsgehalt der gesamten Aussage ermittelt wird. Im Baummodell bedeutet dies, dass eine Ebene nach der anderen vollständig berechnet wird, wobei mit der untersten begonnen wird. Dadurch ist die Eingabe jeder beliebigen 385 Aussage mit bis zu acht Elementaraussagen möglich.

Als Grundlage für die Berechnung dient die eingegebene Aussage des Benutzers. Eine Herausforderung ist dabei, aus den Logikgattern und Elementaraussagen, die dem System als 3-Bit-Werte vorliegen, die auszuwählen, die gerade für die Berechnung benötigt werden. Die Auswahl der Logikgatter erfolgt durch zwei hintereinander geschaltete Multiplexer. Der erste Multiplexer wählt 390 aus den User-Eingängen das aktuell benötigte Logikgatter aus. Das 3-Bit-Signal, das durch den ersten Multiplexer geleitet wurde, wird schließlich als Steuerungssignal für den zweiten Multiplexer verwendet. Dieser verwendet das ausgegebene Signal als Steuerung, um das entsprechende Logikgatter mit dem Zwischenspeicher Z zu verbinden. Der erste Multiplexer generiert also die aktuell benötigten Adressen und der zweite interpretiert diese (vgl. Abbildung 7.1.0.2).

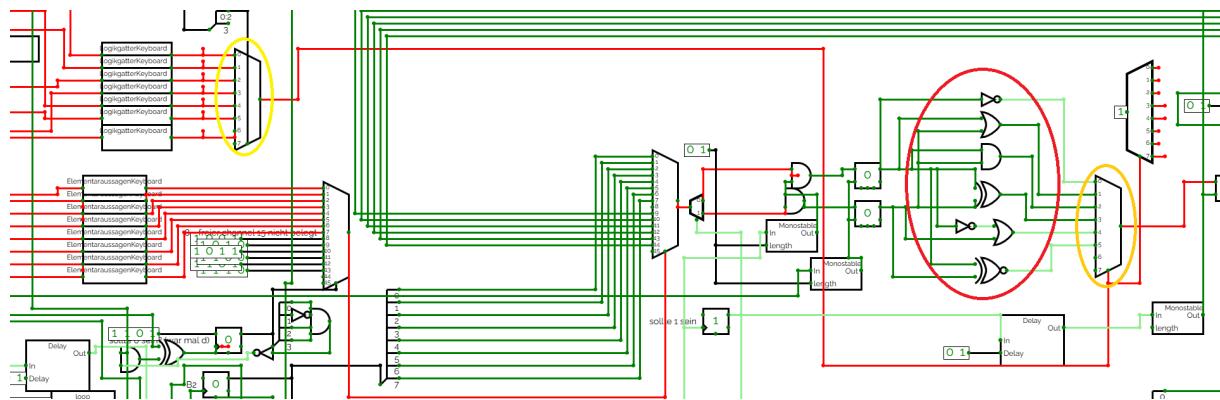


Abbildung 7.1.0.2: Multiplexer 1 (gelb) gibt das zu verwendende Logikgatter an Multiplexer 2 (orange) weiter, der eins der Logikgatter (rot) mit dem Zwischenspeicher verbindet. Screenshot einer eigens erstellten Schaltung; © CircuitVerse

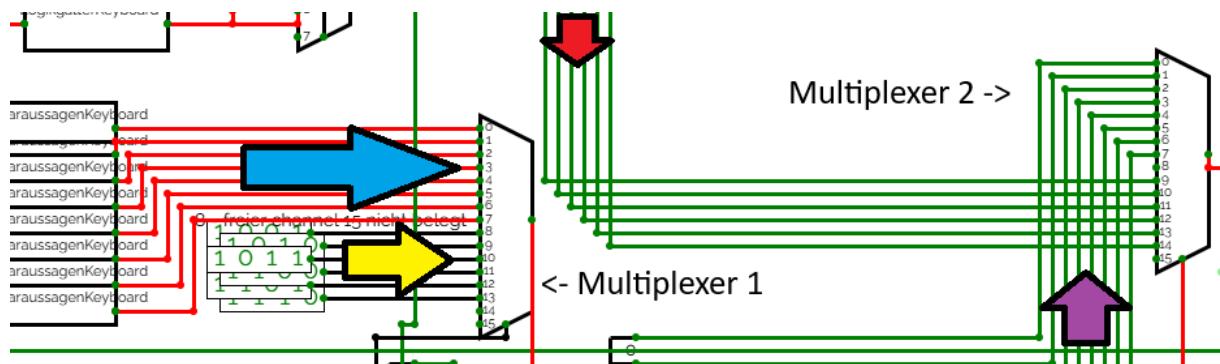


Abbildung 7.1.0.3: Multiplexer zur Auswahl der Aussagenwerte. Screenshot einer eigens erstellten Schaltung; © CircuitVerse

- 395 Wie man anhand von Abbildung 7.1.0.3 erkennen kann, wird dasselbe Konzept zur Auswahl
der Wahrheitswerte verwendet, bloß wird hier für den ersten Multiplexer eine Bitweite von
vier benötigt. Die Pfeile stellen dabei die Signalflüsse zwischen den Multiplexern dar. Die ersten
acht dieser Werte sind die Zustände der ausgewählten Elementaraussagen A-H. Dafür werden die
Ausgänge des Binärzählers B mit den ersten acht Eingängen des zweiten Multiplexers verbunden
400 (lila). Für die letzten drei Logikgatter werden statt Elementaraussagen die im Zwischenspeicher
Z berechneten Werte verwendet, die mit den übrigen Eingängen des zweiten Multiplexers ver-
bunden sind (rot).

Die ersten acht Eingänge des ersten Multiplexers sind, wie bei den Logikgattern, mit der Tastatureingabe verbunden (blau). Durch die übrigen Eingänge werden konstante Zahlenwerte 405 übertragen, die den Zugriff auf die korrekten Werte des Zwischenspeichers durch den zweiten Multiplexer steuern (gelb). Um das korrekte Flip-Flop zum Zwischenspeichern auszuwählen, erfolgt die Verbindung durch einen Demultiplexer, der dem Signal, das von den Logikgattern kommt, den korrekten Speicherplatz zuweist.

⁴¹⁰ Zur Berechnung eines einzelnen Logikoperators werden zunächst die zwei benötigten Wahrheitswerte in je einem S-R-Flip-Flop zwischengespeichert. Wurden alle Gatter und Aussagen zugewiesen, wird der Ausgang des Logikgatters in einem der S-R-Flip-Flops des Zwischenspeichers Z festgehalten. In dem Beispielbaum aus Abbildung 7.1.0.1 würde das bedeuten, dass die Werte von A und B (z.B. 1 und 0) an das „ \rightarrow “-Logikgatter weitergegeben werden und der Ausgang 0 ⁴¹⁵ zwischengespeichert wird.

Nachdem auf diese Weise alle Ebenen des Baums berechnet wurden, wird über ein Signal der nächste Durchgang gestartet.

7.2 Koordination der Abläufe

Die Auswahl der aktuell benötigten Werte und Operatoren erfolgt durch eine Schleife, die bei jeder Wiederholung an verschiedenen Stellen Zähler aktiviert. Diese Zähler steuern jeweils die zwei ⁴²⁰ vorderen Multiplexer zur Auswahl der Logikgatter und Aussagen sowie den Demultiplexer vor dem Zwischenspeicher. Während zur Auswahl der Wahrheitswerte nach jeder Iteration weitergezählt werden muss, muss dies für die Auswahl des Logikgatters und des Zwischenspeicher-Flip-Flops nur nach jedem zweiten Durchgang geschehen, da ein Gatter jeweils zwei Wahrheitswerte ⁴²⁵ verknüpft. Insgesamt gibt es in dem Zyklus von zwei Wiederholungen der Schleife folgende verschiedene Zeitpunkte, zu denen teils mehrere Prozesse gleichzeitig ablaufen:

1. Auswahl des Logikgatters und des ersten Wahrheitswertes über die Multiplexer, Zurücksetzen der zwei S-R-Flip-Flops vor den Logikgattern
2. Zwischenspeichern des ersten Wahrheitswertes im ersten der beiden S-R-Flip-Flops
- ⁴³⁰ 3. Auswahl des nächsten Wahrheitswertes über die Multiplexer
4. Zwischenspeichern des zweiten Wahrheitswertes im zweiten S-R-Flip-Flop
5. Auswahl des nächsten Platzes im Zwischenspeicher Z
6. Speichern des Wertes im Zwischenspeicher Z

Nachdem alle sechs Schritte für ein Logikgatter abgeschlossen sind, wird der Zyklus für das ⁴³⁵ nächste Gatter gestartet. Daher wiederholen sie sich für jedes Logikgatter und jede Zeile der Tabelle. Diese Prozesse müssen so koordiniert werden, dass es zwischen jedem Schritt genug Zeit für die Logikgatter gibt, sich zu aktualisieren. Ansonsten kommt es zu Fehlern, da teils noch nicht berechnete Zwischenergebnisse weiterverwendet werden können. Da es außerdem bei einer ständigen Verbindung mit den zwei S-R-Flip-Flops und dem Zwischenspeicher zu einem ⁴⁴⁰ Speichern von falschen Ergebnissen kommen könnte, werden die Verbindungen standardmäßig

gesperrt und nur zum richtigen Zeitpunkt durch die Aktivierung eines AND-Gatters, der Durchgang des Signals ermöglicht. Der gesamte Prozess zur Berechnung einer Zeile ist in Abbildung 7.2.0.1 grafisch dargestellt.

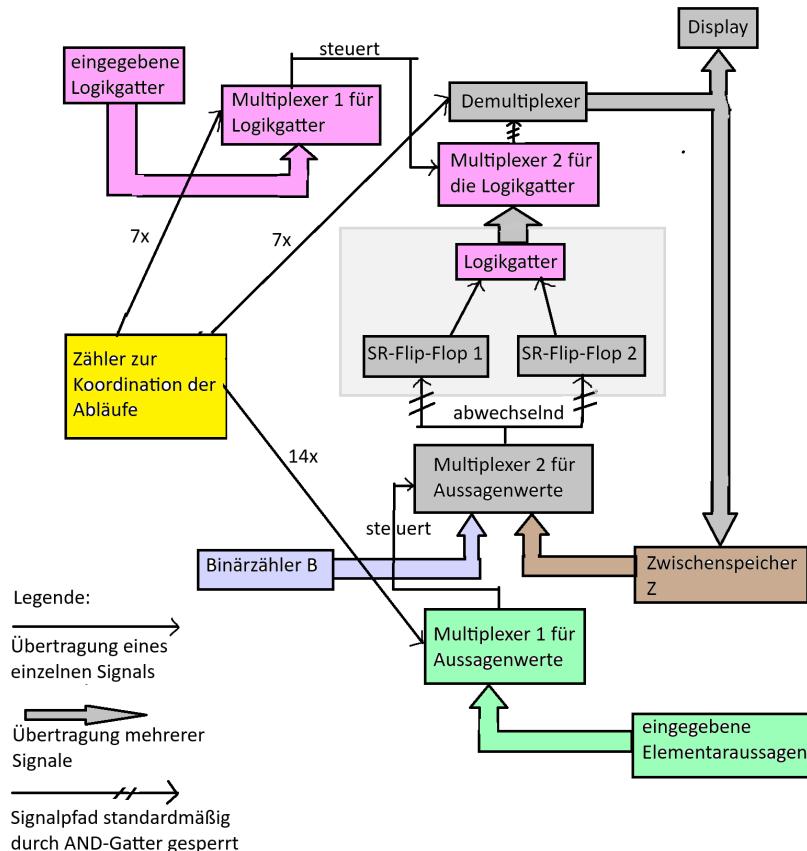


Abbildung 7.2.0.1: Flussdiagramm zur Berechnung einer Tabellenzeile. Eigene Darstellung.

445 7.3 Zentrale Steuerung

Diese Berechnung muss für jede Kombination von Wahrheitswerten der Elementaraussagen durchgeführt werden. Sei n die Anzahl der verwendeten Elementaraussagen, so folgt, dass 2^n Durchgänge gebraucht werden, um alle Kombinationen zu berücksichtigen (Beckermann, A., 2014, S.86). Dazu wird pro Berechnung einer Zeile der Tabelle ein Puls an den Binärzähler B gesendet, wodurch die nächste Kombination an Wahrheitswerten für die Aussagen A-H abgerufen wird. Dabei wird auch der Zwischenspeicher Z und die Binärzähler zur Auswahl der Wahrheitswerte und Logikgatter zurückgesetzt. Da nicht jedes Mal die maximalen 2^8 Durchgänge benötigt werden, wird die Berechnung genau dann abgebrochen, wenn die n -te Stelle des Zählers B aktiviert und wieder deaktiviert wird, da dies nach genau 2^n Wiederholungen geschieht. Realisiert

- 455 wird dies durch einen Multiplexer, dessen Kontrollsignal die Zahl der verschiedenen verwendeten Elementaraussagen ist und dessen Eingänge je eine Stelle des Zählers sind. Das führt dazu, dass der Prozess nach den erwünschten 2^n Wiederholungen gestoppt wird.

7.4 Benutzerschnittstelle

Der Benutzer kann die Aussagen und Logikoperatoren an einem Nachbau der Baumstruktur 460 (vgl. Abbildung 7.1.0.1) auswählen. Im Falle eines NOT-Gatters, das nur einen Wahrheitswert zur Berechnung benötigt, wird nur der linke der beiden Werte weiterverwendet, die sich in der Ebene unter dem Gatter befinden. Will der Benutzer nur $n < 8$ Aussagen verwenden, kann er dies angeben, wodurch die Zeit der Berechnung deutlich reduziert wird⁷. Damit in diesem Fall 465 immer noch alle Möglichkeiten berechnet werden, sollten dabei die ersten n Buchstaben des Alphabets verwendet werden. Das System kann auch eine korrekte Wahrheitstabelle erzeugen, wenn nur ein Teil des Baumes ausgefüllt wurde, insofern man die Resultate an den richtigen Stellen abliest. Für nicht verwendete Gatter bzw. Aussagen wird standardmäßig mit dem Wert „Falsch“ gerechnet.

Die Ausgabe der Werte erfolgt über ein Display, bei dem ein aktiver Pixel für „wahr“ steht. 470 Details über den konkreten Umgang mit den Ein- und Ausgabestellen in den beiden Modellen sind Kapitel B.4 des Anhangs zu entnehmen.

8 Test und Evaluation

8.1 Test der beiden Modelle

Zum Test der Modelle soll zunächst die Aussage X aus Abbildung 7.1.0.1 von den Systemen berechnet werden⁸. Diese ist gut geeignet, da sie alle möglichen Operatoren (mit der Ausnahme des NOT-Gatters) und Aussagen enthält. Sie prüft jedoch nicht, ob beliebige Aussagen an beliebigen Stellen verwendet werden können, da A-H in aufsteigender Reihenfolge verwendet werden. Aus diesem Grund soll als Zweites die Aussage $Y : \neg B \vee (B \wedge A)$ geprüft werden. Hier werden nur zwei Elementaraussagen verwendet, wodurch man das frühere Abbrechen des Generators testen kann. 480 Außerdem werden die Elementaraussagen doppelt und in verschiedener Reihenfolge verwendet sowie der NOT-Operator getestet. Um den Grenzfall mit nur einer verwendeten Elementaraussage zu testen, wird als dritte Aussage Z noch $A \leftrightarrow A$ geprüft. Um die Vergleichbarkeit der Laufzeiten zu gewährleisten, sollten die Clock-Intervalle zwischen den beiden Systemen gleich

⁷vgl. Kapitel 7.3

⁸zur Erinnerung: $(A \rightarrow B) \wedge (C \leftrightarrow D) \vee (E \vee F) \oplus (G \wedge H)$

sein. Dazu wurde die normale Tick-Rate⁹ in Minecraft von 20 auf 40 Hz verdoppelt¹⁰, wodurch
485 Redstone-Ticks alle 50ms verarbeitet wurden. Dies entspricht dem kürzesten Clock-Intervall, was in CircuitVerse einstellbar ist. Dadurch können Unterschiede in der Implementierung, insbesondere die Effizienz der Steuerungsschleifen, objektiv verglichen werden.

Die drei Aussagen wurden jeweils einmal vollständig berechnet, wobei ein Timer bei Betätigung
490 des Start-Knopfes gestartet und mit Abschluss der letzten Berechnung des Systems gestoppt wurde. Da der Prozess vor allem durch konstant zeitlich verzögerte Schleifen koordiniert wird, werden im Modell für die gleiche Zahl an verwendeten Aussagen immer gleich viele Berechnungsschritte durchgeführt und, zumindest in der Simulation, immer dieselbe Zeit benötigt. Daher kann die einzige Art der Abweichung nur auf Messungenauigkeiten oder Schwankungen in der Rechnerleistung zurückzuführen sein. Um dies auszuschließen, wurde in Minecraft geprüft, dass die
495 Tickrate tatsächlich 40Hz betrug, was der Fall war¹¹. Um sicherzugehen, dass die Ergebnisse replizierbar sind, wurde für die Aussage Y insgesamt jeweils viermal die Zeit gestoppt. Dabei lag die gemessene Berechnungszeit sowohl in Minecraft als auch in CircuitVerse jedes Mal im Bereich von ± 1 s.

Aus den Messwerten soll außerdem eine Formel zur benötigten Berechnungszeit in Abhängigkeit
500 von der Zahl der verwendeten Elementaraussagen hergeleitet werden. Da die Berechnung einer Zeile der Wahrheitstabelle konstant viel Zeit einnimmt, gilt: $T(z) = m * z + T_0$ mit T_0 als Zeit, die das System zu Beginn und am Ende der Berechnung zusätzlich benötigt und m als Rechenzeit pro Zeile der Wahrheitstabelle. Da beim Stopp des Systems der erste Schritt doppelt ausgeführt wird, gilt für die Zahl der zu berechnenden Zeilen $z = 2^n + 1$ mit n als Zahl der verschiedenen
505 Elementaraussagen (vgl. Kapitel 7.3). Die finale Formel lautet daher:

$$T(n) = m * (2^n + 1) + T_0$$

8.2 Ergebnisse und Auswertung

Formel	T_{CV} in s (mm:ss)	T_{MC} in s (mm:ss)	$T_{MC} \div T_{CV}$
Aussage X	2780 (46:20)	7298 (121:38)	2,63
Aussage Y	55	145 (2:25)	2,63
Aussage Z	32	89 (1:29)	2,78

Tabelle 7: Berechnungszeiten und Verhältnisse zwischen CircuitVerse (links) und Minecraft (rechts)

⁹Ein Tick ist mit 50ms das kleinste Zeitintervall in Minecraft und ist damit halb so lang wie ein Redstone-Tick.

¹⁰Dies ist durch Eingabe des Commands `/tick rate 40` in die in Minecraft integrierte Befehlszeile möglich

¹¹Dies lässt sich durch Eingabe des Commands `/tick query` in die in Minecraft integrierte Befehlszeile prüfen, die benötigte Rechenzeit pro Tick lag sogar deutlich unter dem festgelegten Intervall von 25ms, wodurch extreme Schwankungen in der Rechnerleistung höchst unwahrscheinlich sind.

Die Ergebnisse können der Tabelle 7 entnommen werden. Die Wahrheitstabellen für Y und Z wurden vollständig, die von X stichprobenartig händisch überprüft, wobei keine Fehler festgestellt werden konnten. Ausschnitte der Ergebnistabellen für die Aussage X sind Abbildung 510 B.4.0.3 zu entnehmen. Um m zu bestimmen, erhält man aus Umstellen der Formel nach T_0 und Gleichsetzen nach einigen Umformungen folgende Gleichung:

$$m = \frac{T(n_1) - T(n_2)}{(2^{n_1} + 1) - (2^{n_2} + 1)}$$

Durch Einsetzen der Werte für die Aussagen X und Z erhält man:

$$\text{CircuitVerse: } m_{CV} = \frac{2780\text{s} - 32\text{s}}{(2^8 + 1) - (2^1 + 1)} \approx 10,8\text{s}$$

$$\text{Minecraft: } m_{MC} = \frac{7298\text{s} - 89\text{s}}{(2^8 + 1) - (2^1 + 1)} \approx 28,4\text{s}$$

Durch weitere Umformungen lässt sich T_0 berechnen:

$$T_0 = T(n) - m * (2^n + 1)$$

515 Durch Einsetzen der Werte der Aussage Z erhält man:

$$\text{CircuitVerse: } T_{0CV} = 32\text{s} - 10,8\text{s} * (2^1 + 1) \approx -0.4\text{s}$$

$$\text{Minecraft: } m_{MC} = 89\text{s} - 28,4\text{s} * (2^1 + 1) \approx 3,8\text{s}$$

Die -0,4s sind lediglich ein Ergebnis der Messungenauigkeit, da CircuitVerse die Berechnung der Tabellenzeilen sofort startet. Dadurch beträgt T_0 wie erwartet ungefähr 0s. In Minecraft gibt es außerdem die Möglichkeit, die Tickrate zu beschleunigen¹². Der Grad der Beschleunigung ist 520 von der Rechnerleistung abhängig, jedoch ist zusätzlich eine Formel für die Berechnungszeit mit 773 statt 40 Ticks pro Sekunde angegeben. Die 773 Ticks wurden dabei bei Ausführung einer beschleunigten Berechnung des Tabellengenerators gemessen. Dadurch wird die Berechnungszeit um den Faktor $\frac{40}{773}$ gekürzt. Die Formeln lauten also:

$$\text{CircuitVerse: } T_{CV}(n) = 10,8\text{s} * (2^n + 1)$$

$$\text{Minecraft (40 Ticks pro Sekunde): } T_{MC1}(n) = 28,4\text{s} * (2^n + 1) + 3,8\text{s}$$

525

$$\text{Minecraft (773 Ticks pro Sekunde): } T_{MC2}(n) = \frac{40}{773} * (28.4\text{s} * (2^n + 1) + 3.8\text{s})$$

¹²Dies ist durch Eingabe des Commands `/tick sprint 3000d` in die in Minecraft integrierte Befehlszeile möglich

Die Berechnungszeiten sind in Abbildung 8.2.0.1 grafisch dargestellt. Die Formeln prognostizieren für $n = 2$ eine Zeit von ca. 54 bzw. 146 Sekunden, was in etwa den gemessenen Zeiten der Aussage Y entspricht.

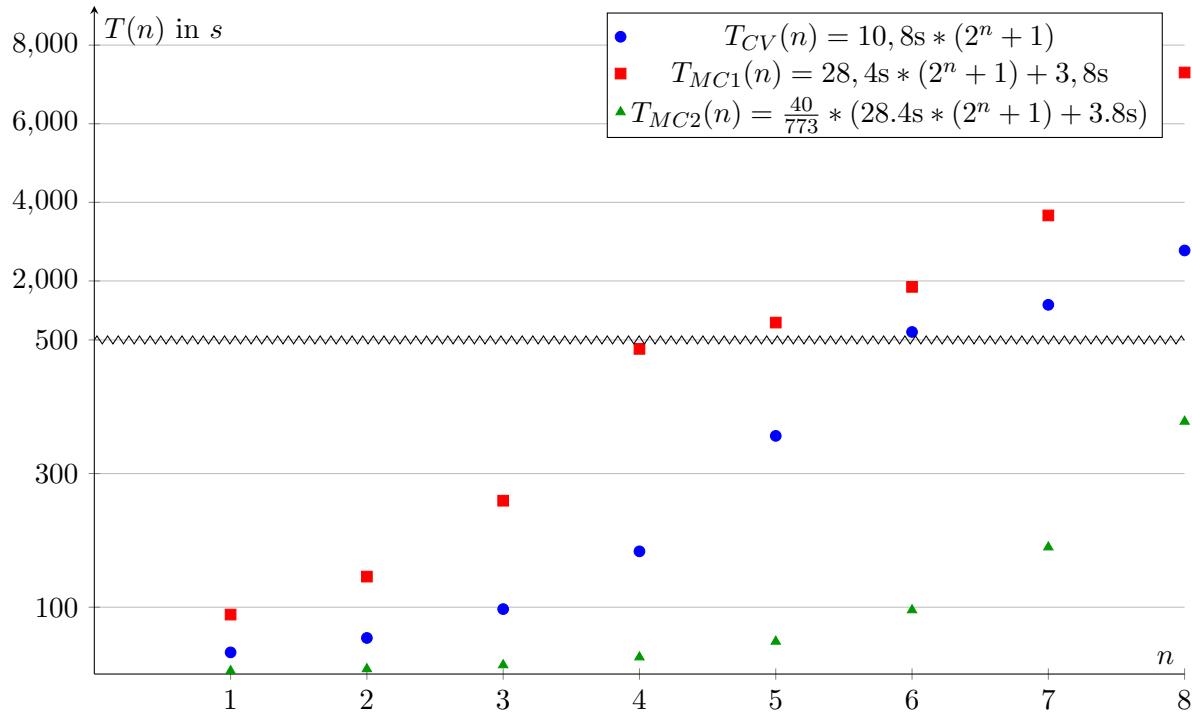


Abbildung 8.2.0.1: Laufzeiten der Modelle in Abhängigkeit der Zahl der verwendeten Elementaussagen n . Ergebnisse wurden zur besseren Übersicht ab $T = 500$ s skaliert. Eigene Darstellung.

Rechnet man das Verhältnis $\frac{T_{MC}}{T_{CV}}$ der Zeiten zwischen den beiden Systemen aus, lässt sich feststellen, dass Circuit Verse für größere Werte von n um den Faktor 2,63 schneller ist als Minecraft. Dieser Zusammenhang zeigt sich auch, wenn man mithilfe des Grenzwertes das Verhältnis zwischen den beiden Formeln bildet:

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{T_{MC1}(n)}{T_{CV}(n)} &= \lim_{n \rightarrow \infty} \frac{28.4s \cdot (2^n + 1) + 3.8s}{10.8s \cdot (2^n + 1)} = \lim_{n \rightarrow \infty} \frac{28.4s \cdot (2^n + 1)}{10.8s \cdot (2^n + 1)} + \frac{3.8s}{10.8s \cdot (2^n + 1)} = \\ &= \frac{28.4s}{10.8s} + \lim_{n \rightarrow \infty} \frac{3.8s}{10.8s \cdot (2^n + 1)} = 2.63 + 0 = 2.63 \end{aligned}$$

Die kleine Abweichung des Verhältnisses bei Aussage Z kommt daher, dass $\frac{3.8s}{10.8s \cdot (2^n + 1)}$ für kleine Werte von n noch deutlich über 0 liegt.

Die Unterschiede in der Geschwindigkeit haben zwei Gründe: Erstens wurde der Minecraft-Wahrheitstabellengenerator zuerst konstruiert, wodurch bei CircuitVerse manche Prozesse optimiert werden konnten. Zweitens arbeiten die Bauteile in CircuitVerse praktisch ohne Zeitverzögerung, während jedes elementare Schaltkreis-Element in Minecraft, wie in etwa Verstärker

540 oder Fackeln, mindestens ein Clock-Intervall Verzögerung mit sich bringen. Diese Verzögerung summiert sich durch Signalübertragung durch lange Leitungen, sowie komplexere Schaltkreise wie Multiplexer, die aus zahlreichen einzelnen Bauteilen bestehen.

Beide Modelle sind also im Rahmen der getesteten Wahrheitstabellen voll funktionstüchtig, jedoch gibt es in CircuitVerse einen nicht behebbaren Software-Fehler, der die Berechnung 545 verfälschen kann. Die Ursache des Fehlers ist vermutlich, dass Sub-Circuits nur unvollständig geladen werden, wodurch diese teils falsch ausgeführt oder beim Interagieren mit völlig anderen Elementen ausgelöst werden. Das kann beispielsweise dazu führen, dass zu Beginn der Berechnung Schleifen nicht oft genug wiederholt werden, wodurch die ersten Zeilen der Wahrheitstabelle falsch sein können. Das System setzt sich jedoch nach jedem Durchgang von selbst zurück, wes- 550 halb sich der Fehler von selbst behebt. Alternativ kann die Berechnung nach dem ersten Durchlauf abgebrochen und neu gestartet werden. Dadurch werden Werte, die sonst möglicherweise falsch berechnet sein könnten, korrigiert.

8.3 Diskussion: Einordnung in die Von-Neumann-Architektur

Da das Ziel dieser Arbeit ist, die Architektur von Computern durch Logikschaltkreise nach- 555 zu vollziehen, soll nun geprüft werden, inwiefern der Wahrheitstabellengenerator der Definition eines Computers entspricht.

Viele Rechner basieren auf der sogenannten Von-Neumann-Architektur. Diese beschreibt die Unterteilung eines Computers in einen Prozessor, ein Ein- bzw. Ausgabesystem und einen Speicher, der sowohl Programme als auch Daten speichert. All diese Elemente kommunizieren über 560 ein Bussystem (Hellmann, 2022, S. 46).

Im Wahrheitstabellengenerator sind Ansätze dieser Strukturen zu erkennen. Die sieben Logikgatter sind eine Art von Rechenwerk, da hier die ergebnisrelevanten mathematischen Operationen zur Berechnung der Wahrheitswerte durchgeführt werden. Als Steuerwerk könnte das System aus Multiplexern inklusive ihrer Ansteuerung (vgl. Kapitel 7.2) interpretiert werden, da 565 es die Berechnungen des Systems koordiniert. Rechenwerk und Steuerwerk bilden in der Von-Neumann-Architektur den Prozessor.

Es liegt auch ein E/A-System vor, da der Benutzer eine Aussage eingeben und über ein Display ablesen kann und es gibt einen Zwischenspeicher, der jedoch nicht Programme, sondern lediglich Daten in Form von Zwischenergebnissen speichert.

570 Ein weiteres Merkmal von Computern ist die Programmierbarkeit, was bedeutet, dass man „frei bestimmen [kann], welche Operationen in welcher Reihenfolge durchgeführt werden“ (Hellmann, 2022, S. 5f). Dies ist beim Tabellengenerator nicht gewährleistet, da er nur für das Generieren

von Wahrheitstabellen geeignet ist.

Insgesamt ist der Aufbau des Generators also mit den grundlegenden Prinzipien der Von-Neumann-Architektur vergleichbar, jedoch ist die Definition eines Computers nicht vollständig erfüllt. Für eine vollständige Nachbildung eines Computers müsste der Generator programmierbar gestaltet werden, sodass beliebige Operationen in variabler Reihenfolge durchgeführt werden können.

9 Fazit

Insgesamt ist also deutlich geworden, wie tief grundlegende Prinzipien der Logik in digitalen Systemen verankert sind. Dazu wurde nachgewiesen, dass selbst die Existenz primitivster Logikgatter, wie Leitungen und Fackeln in Minecraft, die Basis für die Konstruktion komplexer Prozessoren sind. Das hat sich auch darin gezeigt, dass sich das in Minecraft entwickelte Schema des Wahrheitstabellengenerators problemlos auf realitätsnähere Schaltkreissimulationen wie CircuitVerse übertragen ließ.

Hinsichtlich der Frage, welche minimalen logischen Grundbausteine zur Konstruktion von Computern ausreichen, konnte festgestellt werden, dass zumindest für eine Vorstufe eines Rechners lediglich ein NOR- oder NAND-Gatter und eine praktikable Methode zur Zeitverzögerung nötig sind. Letztere kann, je nach Modell, auch nur durch Logikgatter umgesetzt werden.

Eine Möglichkeit zur Weiterentwicklung wäre, die aktuell für größere Wahrheitstabellen relativ lange Berechnungszeit zu kürzen. In Minecraft wäre das durch eine effizientere Taktung umsetzbar. Ein Ansatz für beide Systeme wäre, Parallelität zu ermöglichen und so mehrere Zwischenergebnisse auf einmal zu berechnen.

Ein letzter Schritt zum Nachweis der Übertragbarkeit des Systems wäre die Konstruktion eines Rechners basierend auf der Von-Neumann-Architektur. Aufgrund seiner universellen Programmierbarkeit könnte auf einem solchen Rechner nicht nur der Wahrheitstabellengenerator simuliert werden, sondern – vorausgesetzt, genügend Ressourcen sind verfügbar – auch jedes andere Programm, das auf Rechnern ausgeführt werden kann. So wäre es theoretisch möglich, in Minecraft selber eine Simulation von CircuitVerse zu starten und in dieser den hier besprochenen Wahrheitstabellengenerator zu konstruieren. Dies alles würde wiederum auf einem realen PC ausgeführt werden – einem System, das selbst im Kern nichts anderes als eine hochgradig optimierte Zusammensetzung logischer Funktionen darstellt.

A Anmerkungen zur Konstruktion logischer Schaltungen in den beiden Modellen

605 A.1 Vereinfachungen durch Minecraft-Mechaniken

Kompakte AND-Gatter Mithilfe eines sogenannten Kolbens (minecraft:redstone_tube) lassen sich AND-Gatter deutlich platzsparender konstruieren. Dieser bewegt bei Aktivierung einen benachbarten Block von sich weg und kann diesen nach Deaktivierung wieder zurückziehen. Durch eine Konstruktion wie in Abbildung A.1.0.1, in der eine zweite Redstone-Leitung durch einen vom Kolben bewegten Block geleitet wird, kommt das Signal nur durch, wenn der Kolben ausgefahren und die ursprüngliche Leitung aktiviert ist.

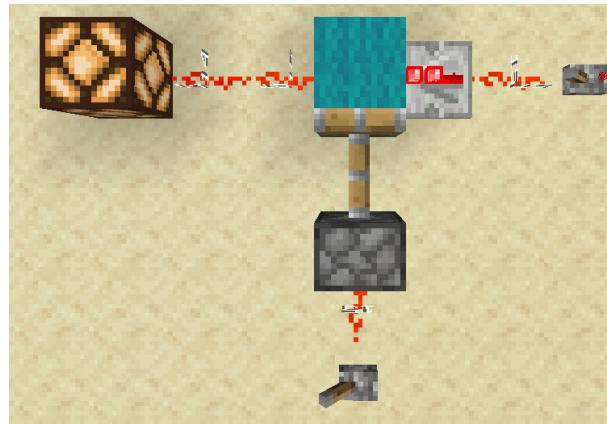


Abbildung A.1.0.1: Kompaktes AND-Gatter, bei dem aktuell beide Ausgänge aktiviert sind. Eigene Darstellung; © für Texturen: Mojang Studios

Signalstärke Die Signalstärke von Redstone-Leitungen kann zwar ein Hindernis sein, da sie die Reichweite von Kabeln limitiert, jedoch kann sie auch genutzt werden, um Signale mit einer Bitbreite von bis zu vier Bits zu speichern und zu übertragen. Die Signalstärke hat nämlich 615 16 verschiedene Zustände (und entspricht somit $\log_2(16) = 4$ Bits), die sich in binäre Signale übersetzen lassen. Wird z.B. für den Eingang eines Decoders bzw. Multiplexers eine 4-Bit-Zahl benötigt, kann man folglich diese Information über eine einzige Redstone-Leitung übertragen.

Signalstärke-Decoder und Multiplexer Die Umwandlung der Signalstärke wird durch einen Signalstärke-Decoder umgesetzt, der wie in Abbildung A.1.0.2 dargestellt funktioniert. 620 Die weißen Bauteile (minecraft:redstone_comparator) sind sogenannte Redstone-Komparatoren. Diese geben das Signal, welches sie als Eingang erhalten, unverändert weiter¹³, außer es liegt ein Signal an der Seite an. In diesem Fall wird diese Signalstärke vom originalen Signal subtrahiert. Diese Funktion wird

¹³Dies ermöglicht auch die Übertragung einheitlicher Signalstärken über längere Entfernungen



Abbildung A.1.0.2: Redstone-Signalstärke-Decoder. Eigene Darstellung; © für Texturen: Mojang Studios

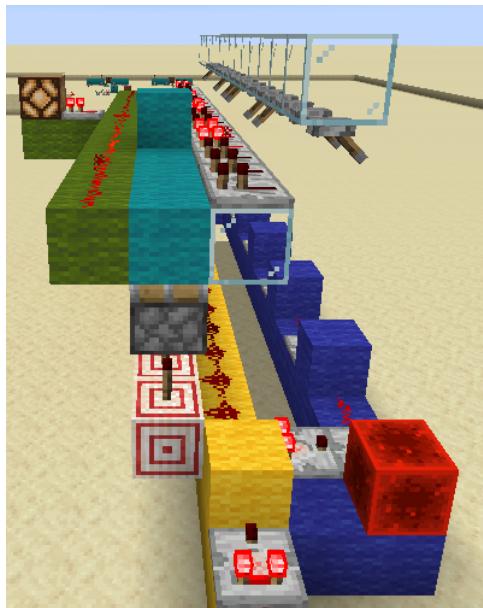


Abbildung A.1.0.3: Erweiterung eines Decoders zum Multiplexer (Seitenansicht). Eigene Darstellung; © für Texturen: Mojang Studios

genutzt, um das Eingangs-Signal einmal unverändert durch den Decoder zu speisen, während parallel ein anderes Signal von der gegenüberliegenden Seite entgegenkommt. Der weiß eingekreiste Komparator subtrahiert die Eingangs-Signalstärke von 15, woraufhin diese über die blaue Leitung unverändert ans andere Ende weitergeleitet wird. Dadurch bleibt immer genau einer der 16 Blöcke ohne Signal, aus dem man dann mithilfe einer Redstone-Fackel ein Signal auslesen kann¹⁴. Nutzt man die Ausgänge des Decoders, um mit AND-Gattern mehrere Signale zu steuern, erhält man einen Multiplexer (vgl. Kapitel 4.4), den man sowohl mit Signalstärke steuern kann als auch Signale mit einer bestimmten Signalstärke übertragen kann

Monostabile Kippstufe Für eine monostabile Kippstufe wird ein Kolben und ein Beobachter (/icon/box/) verwendet. Ein Beobachter ist ein Block, der jedesmal ein Signal ausgibt, wenn sich ein

¹⁴In der Abbildung A.1.0.2 liegt eine Signalstärke von vier an. Da eine Signalstärke von null den ersten Ausgang aktivieren würde, leuchtet hier die fünfte Lampe von links.

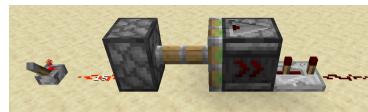


Abbildung A.1.0.4: Monostabile Kippstufe nach Aktivierung. Eigene Darstellung; © für Texturen: Mojang Studios

Nachbarblock ändert. Bewegt man einen solchen Block per Kolben, so gibt der Beobachter ein einmaliges Signal aus, das z.B. durch Redstone-Verstärker beliebig verlängert werden kann.

⁶³⁵ (siehe Abbildung A.1.0.4)

Alternative Arten von Flip-Flops Die sogenannte Kupferbirne (💡) stellt ein kompaktes T-Flip-Flop dar, indem sie einen stabilen Zustand besitzt und diesen jedesmal ändert, wenn sie aktiviert wird. Ob die Lampe aktuell leuchtet oder nicht, kann über einen Redstone-Komparator ausgelesen werden. Dadurch eignet sie sich außerdem als Pixel eines Displays, wie im Anhang

⁶⁴⁰ Kapitel B.4 erläutert wird.

Für S-R-Flip-Flops können zwei Spender (📦) verwendet werden, die gegenseitig miteinander verbunden sind. Ein Spender verschiebt bei Aktivierung einen Gegenstand aus seinem Inventar in einen benachbarten Block. Wird so eine Seite des Schaltkreises aktiviert, wird der Gegenstand in den anderen Spender verschoben und bleibt dort, bis jene Seite aktiviert wird. Welcher der beiden ⁶⁴⁵ Spender gerade einen Gegenstand enthält, kann über einen Redstone-Komparator ausgelesen werden. Werden beide Spender gleichzeitig aktiviert, ist das Verhalten nicht konsistent, was bei einem S-R-Flip-Flop den illegalen Zustand darstellt.



Abbildung A.1.0.5: J-K-Flip-Flop basierend auf Spendern. Eigene Darstellung; © für Texturen: Mojang Studios

Zähler Es ist möglich, durch Kupferbirnen einen Binärzähler zu konstruieren, indem man mehrere von diesen aneinanderreihrt, wie in Abbildung B.1.0.1 zu sehen ist. Dadurch hat man 650 die unter Kapitel 4.2 dargestellte Aneinanderreihung von T-Flip-Flops nachgestellt und kann deren Zustände auf der Seite der Kupferbirnen auslesen.

Da manche Multiplexer im Wahrheitstabellengenerator über Signalstärke angesteuert werden, ist es außerdem hilfreich, einen Zähler zu verwenden, der bei jeder Aktivierung eine Signalstärke um 1 erhöht. Dies kann man mit dem Schaltkreis aus Abbildung A.1.0.6 erreichen: Dessen 655 Grundlage ist eine Rückkopplung aus zwei Redstone-Komparatoren (grün), die ein Signal mit einer bestimmten Stärke speichern. Um das Signal um 1 zu erhöhen, wird die Signalstärke via einen weiteren Redstone-Komparator von 15 subtrahiert und über zwei Blöcke weitergeleitet. Dadurch verringert sich die Stärke um 1, woraufhin dieses Signal erneut von 15 subtrahiert wird. (gelb) Dadurch erhält man die ursprüngliche Signalstärke um 1 erhöht. Verbindet man dieses 660 Signal durch die Aktivierung der blauen Leitung wieder mit der ursprünglichen Signalstärke, zählt der Zähler um 1 hoch. Um den Zähler auf 0 zurückzusetzen, kann man über die weiße Leitung 15 von der aktuellen Signalstärke subtrahieren.

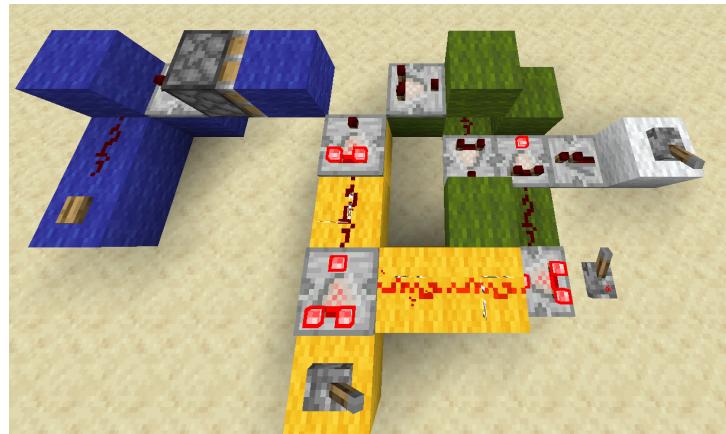


Abbildung A.1.0.6: Signalstärke-Zähler, abgewandeltes Design nach Properinglish19, 2013. Eigene Darstellung; © für Texturen: Mojang Studios

Lesepulte Ist ein Buch auf einem Lesepult befestigt, kann ein Spieler dieses anschauen. Die aktuelle Seite des Buches kann über einen Redstone-Komparator in Signalstärke umgewandelt 665 werden, wodurch es sich als Benutzereingang eignet.

A.2 Sub-Circuits in CircuitVerse

Verzögerung Der in Abbildung B.4.0.2 dargestellte Schaltkreis gibt eine Änderung am Eingangssignal mit einer festlegbaren Verzögerung an den Ausgang weiter. Durch ein XOR-Gatter wird überprüft, ob sich der Eingang aktuell vom Ausgang unterscheidet. Ist dies der Fall, wird

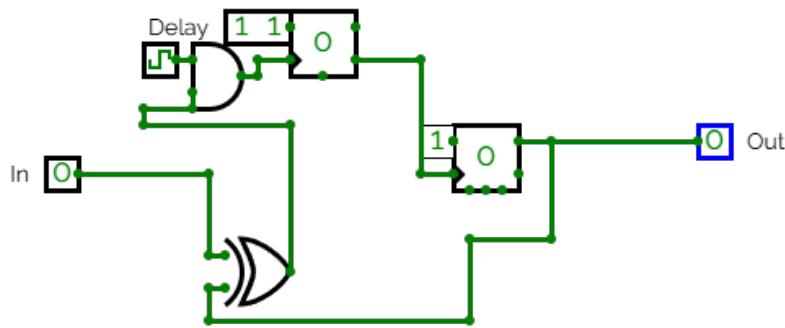


Abbildung A.2.0.1: Verzögerungs-Sub-Circuit. Screenshot einer eigens erstellten Schaltung; © CircuitVerse

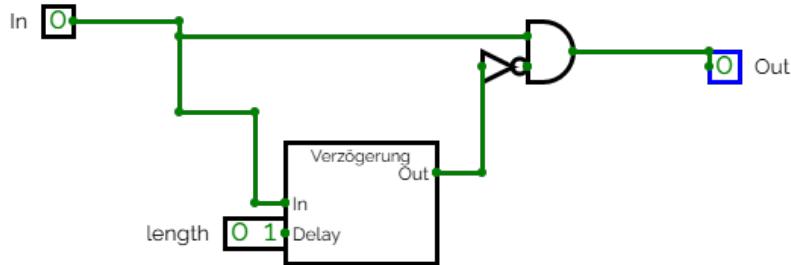


Abbildung A.2.0.2: Monostabile Kippstufe als Sub-Circuit. Screenshot einer eigens erstellten Schaltung; © CircuitVerse

- 670 durch ein AND-Gatter eine Clock mit einem Binärzähler verbunden, dessen Höchstwert über den Delay-Eingang einstellbar ist. Fängt der Binärzähler wieder bei 0 an, schickt er über den Zero-Ausgang ein Signal an den Clock-Eingang eines T-Flip-Flops. Dadurch wird der Zustand des Ausgangs umgeschaltet und durch eine Rückverbindung zum XOR-Gatter wird das Zählen deaktiviert. Dadurch funktioniert der Schaltkreis ähnlich wie ein Redstone-Verstärker (vgl. Kapitel 675 5.1), jedoch mit der Einschränkung, dass es zu Problemen kommen kann, wenn das Eingangs-Signal kürzer als die festgelegte Verzögerung ist. In diesem Fall zählt der Binärzähler nicht zu Ende und das System schaltet beim nächsten Mal schneller als erwünscht. Außerdem kann es zu minimalen Schwankungen der Verzögerung kommen, die davon abhängig sind, zu welchem Zeitpunkt zwischen den einzelnen Clock-Zyklen das System ausgelöst wird.
- 680 **Monostabile Kippstufe** Durch die in Abbildung A.2.0.2 dargestellte Erweiterung des Verzögerungsschaltkreises erhält man ein Äquivalent zur monostabilen Kippstufe. Hier wird das Eingangs-

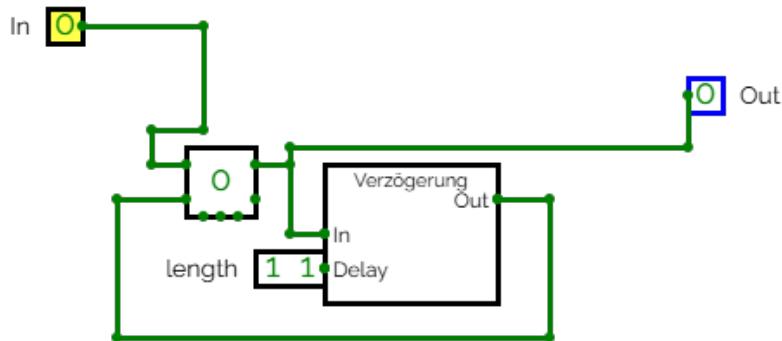


Abbildung A.2.0.3: Schaltkreis zur Signalverlängerung. Screenshot einer eigens erstellten Schaltung; © CircuitVerse

Signal solange an den Ausgang weitergeleitet, bis nach einer festgelegten Verzögerung der Ausgang durch ein AND-Gatter gesperrt wird. Auch hier darf das eingegebene Signal nicht kürzer sein als die Verzögerung, zusätzlich darf der Schaltkreis nicht zu schnell wieder ausgelöst werden, 685 da sonst die Verzögerung immer noch aktiv ist.

Signal-Verlängerung Durch den in Abbildung A.2.0.3 dargestellten Schaltkreis kann ein beliebig kurzes Signal auf eine festgelegte Signallänge verlängert werden. Dazu wird zunächst der S-Eingang eines S-R-Flip-Flops aktiviert. Dessen Ausgang wird wiederum verzögert in seinen R-Eingang geleitet. Ist der Eingang immer noch aktiv, tritt eigentlich der illegale Zustand $S=R=1$ 690 ein, der jedoch zumindest in CircuitVerse dafür sorgt, dass der Ausgang solange aktiv bleibt, bis der S-Eingang wieder 0 ist, was in diesem Fall erwünscht ist. Auch hier sollte der Schaltkreis aufgrund des verspäteten Deaktivieren des Verzögerungs-Subcircuits nicht zu schnell hintereinander aktiviert werden.

Schleifen Bei Eingabe eines Signals wird durch eine monostabile Kippstufe kurz ein S-R-Flip-Flop in den Set-Zustand gesetzt. Dadurch werden wiederholt Signale, deren Länge man über den Eingang „Intervall“ festlegen kann, an einen Binärzähler und an den Ausgang weitergeleitet. Ist der Binärzähler an seinem Limit angekommen, das über „Wiederholungen“ einstellbar ist, beginnt er danach wieder bei 0, gibt deshalb ein Signal mehr aus, als festgelegt wurde und 700 gibt dabei über den Zero-Ausgang ein Signal, das das S-R-Flip-Flop kurz zurücksetzt und den Zustand der Schleife wieder auf inaktiv setzt.

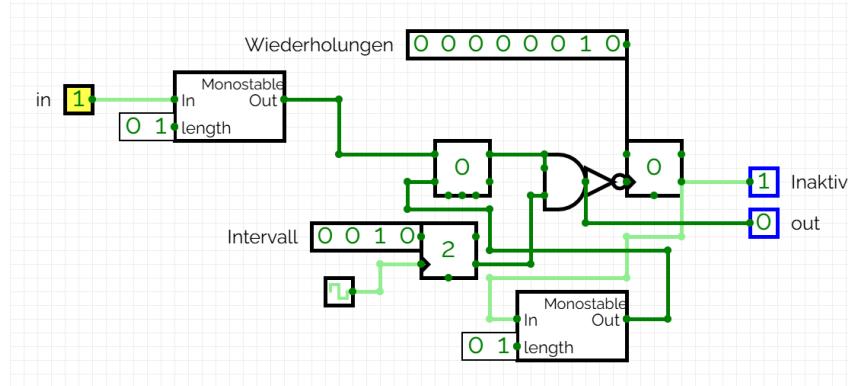


Abbildung A.2.0.4: Sub-Circuit für Schleifen. Screenshot einer eigens erstellten Schaltung; © CircuitVerse

Verarbeitung der Tastatureingabe Um Benutzereingänge für den Wahrheitstabellengenerator zu verwenden, ist zur Verarbeitung der Eingabe ein weiterer Sub-Circuit nötig. Da ein Tastatur-Baustein¹⁵ den ASCII-Wert nur kurz über den Ausgang liefert, wird dieser in einem D-Flip-Flop mit einer ausreichenden Bitweite gespeichert. Dieser Wert dient nun als Steuersignal für einen Multiplexer. Dieser wählt eine 3- bzw. 4-Bit-Zahl aus, die den entsprechenden Binärcode für die gewünschte Aussage bzw. den gewünschten Operator entspricht.

B Unterschiede in der Umsetzung zwischen den beiden Modellen

B.1 Generieren einer Tabellenzeile

⁷¹⁰ **Umsetzung in Minecraft:** Der Binärzähler B1 ist in Minecraft durch einen Kupferleuchten-Zähler¹⁶ implementiert. Von diesem werden auf der einen Seite die Zustände der Elementaraussagen direkt in das Display übertragen und auf der anderen Seite zur Berechnung weiterverwendet. Um Platz zu sparen, wird ausgenutzt, dass im selben Multiplexer Signale von verschiedenen Bitweiten übertragen werden können. Dafür werden zur Auswahl der Wahrheitswerte zunächst die ⁷¹⁵ ersten acht Werte abgerufen und in den zweiten Multiplexer geleitet. Ab dem neunten Wert werden direkt die Zwischenergebnisse aus Z an die Logikgatter weitergeleitet. Dies ist dadurch umgesetzt, dass nur jeweils die ersten acht und danach die übrigen Durchgänge des Multiplexers leitend miteinander verbunden sind, wodurch man je nach Nummer des Eingänge den Ausgang auf zwei verschiedene Arten weiterverwenden kann.

¹⁵Die Funktionsweise eines Tastaturbausteins kann unter CircuitVerse, 2025a nachgeschlagen werden.

¹⁶siehe Anhang Kapitel A.1

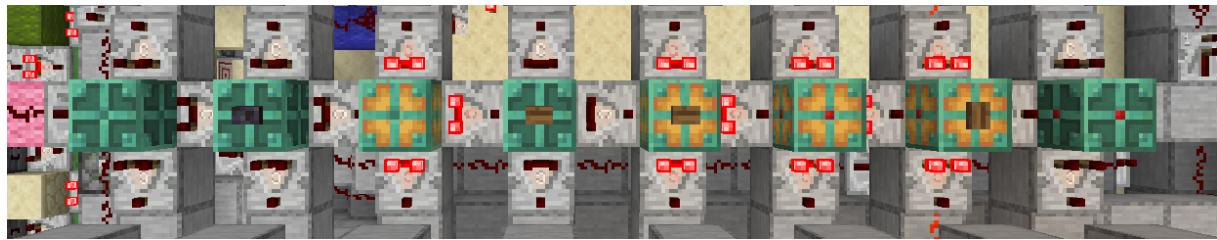


Abbildung B.1.0.1: Binärzähler B1 des Wahrheitstabellengenerators. Eigene Darstellung; © für Texturen: Mojang Studios

- 720 **CircuitVerse** Da CircuitVerse Signale, die mehrere Bits breit sind, standardmäßig in einer Leitung überträgt, muss der aktuelle Wert des Binärzählers durch einen sogenannten Splitter in seine acht Bits zerlegt werden, um die einzelnen Wahrheitswerte abrufen zu können. In CircuitVerse ist es nicht direkt möglich, Signale von verschiedenen Bitbreiten durch denselben Multiplexer zu leiten. Aus diesem Grund sind im ersten Multiplexer nach den ersten acht Elementaraussagen feste Adressen in Form von 4-Bit-Signalen angelegt, die die Stellen im zweiten Multiplexer ansteuern, an denen die Werte des Zwischenspeichers Z anliegen.
- 725

B.2 Koordination der Abläufe

- Minecraft** Die Schleife ist durch eine Clock mit integriertem Zähler realisiert, die in Abbildung B.2.0.1 zu erkennen ist. Diese basiert auf zwei Konstruktionen aus Redstone-Komparatoren, 730 die ähnlich wie Kondensatoren funktionieren. Die beiden Kondensatoren laden und entladen sich abwechselnd, wodurch in regelmäßigen Zeitabständen Impulse ausgesendet werden. Die Clock schickt insgesamt sieben Signale an die Logikgatter. Da die Wahrheitswerte doppelt so oft aktiviert werden müssen, wird das Ausgangssignal in Richtung des entsprechenden Multiplexers verzögert wieder in sich selber gespeist (siehe Pfeile und rotes Rechteck in Abbildung B.2.0.1).

- 735 **CircuitVerse** Im Gegensatz zur Umsetzung in Minecraft wird die Schleife an sich 14-mal aktiviert und dafür gesorgt, dass, wo nötig, nur nach allen zwei Durchgängen ein Signal weitergeleitet wird. Umgesetzt wird dies durch einen 1-Bit-Binärzähler, der nur nach jedem zweiten Puls den Wert 1 weitergibt.

B.3 Zentrale Steuerung

- 740 **Minecraft** Da der für B1 verwendete, auf Kupferleuchten basierende Binärzähler keine integrierte Reset-Funktion besitzt, und das alternative Hochzählen zum Maximalwert zu lange dauern würde, wurde der Zähler um eine Reset-Funktion erweitert. Diese liest den Ausgang einer Stelle aus und verbindet diesen wieder mit dem Eingang des T-Flip-Flops. Werden diese



Abbildung B.2.0.1: Clock zur Koordination der Abläufe. Eigene Darstellung; © für Texturen: Mojang Studios

Leitungen, die davor durch AND-Gatter gesperrt waren, aktiviert, wird jede Stelle, die davor
 745 aktiv war, einmalig aktiviert, wodurch der Zähler auf 0 zurückgesetzt wird. Die übrigen Zähler und der Zwischenspeicher Z können auf herkömmliche Weise zurückgesetzt werden.

CircuitVerse Der zentrale Binärzähler B1 wird zu Beginn über die integrierte Reset-Funktion zurückgesetzt, sodass die Berechnung immer mit der gleichen Kombination startet. Das Zurücksetzen der übrigen Binärzähler und des Zwischenspeichers Z erfolgt durch eine Schleife 750 mit genug Wiederholungen, um theoretisch von 0 bis zum Maximalwert der Zähler hochzuzählen. Dabei wird das Signal zum Zähler nur durchgelassen, wenn er noch nicht seinen Maximalwert m erreicht hat. Dies kann durch einen Decoder umgesetzt werden, dessen m -ter Ausgang mit einem AND-Gatter verbunden ist, durch das das Signal nur durchgelassen wird, wenn es nicht diesem Wert entspricht. Da bei der Berechnung die Zähler schon aktiviert werden, bevor der erste Wert 755 berechnet wurde, sorgt dies dafür, dass die erste Aktivierung alle Zähler auf 0 zurücksetzt und so mit dem ersten Wert gestartet wird.

B.4 Benutzerschnittstelle

Minecraft Die Auswahl der Parameter erfolgt in Minecraft über Lesepulte¹⁷. Soll eine Stelle im Baum nicht belegt werden, kann auf eine leere Seite im Buch des Lesepultes geblättert werden.
 760 Nach Eingabe der Startwerte kann der Prozess über einen Knopf gestartet werden.

Die Pixel des Displays werden durch Kupferleuchten umgesetzt. (vgl. Anhang Kapitel A.1) Dabei

¹⁷ Siehe Anhang Kapitel A.1



Legende:

- Eingabe der zu berechnenden Logikgatter
- Eingabe der Aussagen
- Auswahl der Zahl der verwendeten Elementaraussagen
- Startknopf

Abbildung B.4.0.1: Eingabe des Wahrheitstabellengenerators. Eigene Darstellung; © für Texturen: Mojang Studios

sind jeweils 15 Kupferleuchten in einer Reihe angebracht, die einer Zeile der Wahrheitstabelle entsprechen. Ist die Berechnung einer Zeile abgeschlossen, wird diese durch Kolben nach oben verschoben und die nächste, noch nicht berechnete Reihe von Kupferleuchten rückt nach¹⁸.

⁷⁶⁵ **CircuitVerse** Die Eingabe der Parameter erfolgt über Tastatur-Bausteine¹⁹. Der Baum verläuft von links nach rechts, sodass in der linken Ebene die Aussagen A-H und rechts das letzte Logikgatter eingegeben werden. Für die Logikoperatoren sind die in Tabelle 1 des Hauptdokumentes angegebenen Buchstabenkürzel zu verwenden. Sowohl die Aussagen als auch die Gatter müssen als Kleinbuchstaben angegeben werden. Soll ein Gatter bzw. eine Elementaraussage nicht belegt ⁷⁷⁰ sein, kann ein Minuszeichen eingegeben werden.

Als Display wird in CircuitVerse eine vom Programm bereitgestellte LED-Matrix²⁰ verwendet. Ein Pixel ist aktiv, wenn er weiß leuchtet, lila entspricht dem Wahrheitswert „Falsch“. Die aktuelle Spalte wird über den Wert des Binärzählers B1 gesteuert. Es gibt insgesamt vier dieser Matrizen, je zwei für die Aussagen und die Ergebnisse sowie zwei Schichten übereinander, da die

¹⁸Hinweis: da die Kupferleuchten mechanisch bewegt werden müssen, sind manche Zeilen der Wahrheitstabelle hinter den Blöcken zur Transportation etwas versteckt, können jedoch aus anderen Winkeln ebenfalls betrachtet werden.

¹⁹Deren Signale werden entsprechend Kapitel A.2 des Anhangs weiterverarbeitet

²⁰Siehe CircuitVerse, 2025a

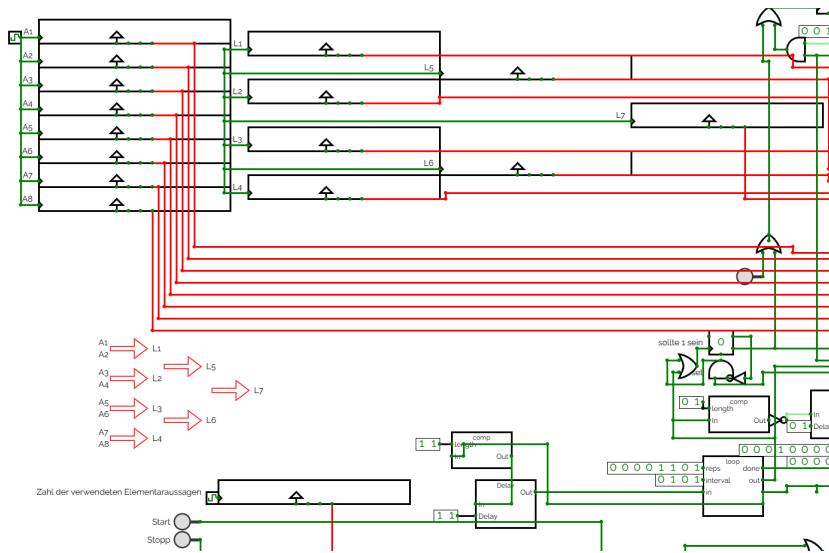
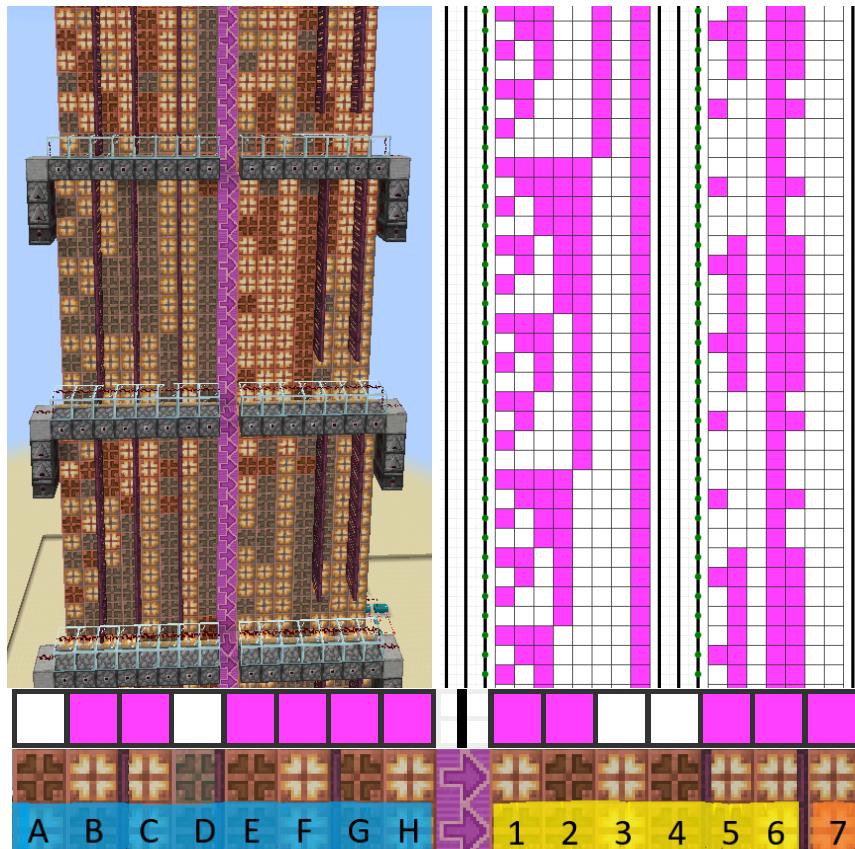


Abbildung B.4.0.2: Benutzerschnittstelle in CircuitVerse. Oben: Nachbau der Baumstruktur; Unten: Eingabe zum Starten der Berechnung. Screenshot einer eigens erstellten Schaltung; © CircuitVerse

- 775 maximale Zeilenanzahl einer Matrix 128 statt der benötigten $2^8 = 256$ beträgt. Ist die Berechnung einer Zeile abgeschlossen, so wird ein Binärzähler durch eine Schleife achtmal aktiviert. Dieser Binärzähler steuert die aktuelle Spalte und zwei Multiplexer, wodurch jeweils die zuvor berechneten Aussagen bzw. (Zwischen-)Ergebnisse in das Display übertragen werden. Der Übergang in die untere Reihe aus Matrizen erfolgt über Demultiplexer, die die Informationen 780 über die aktuelle Zeile, Spalte und Farbe je nach aktueller Zeile an die obere bzw. untere Hälfte überträgt. Das Steuersignal für diese Demultiplexer ist der letzte Bit des Binärzählers B1, da dieser genau ab 129 aktiv ist.



Oben: Ausschnitt des Displays; Unten: Zeile der Wahrheitstabelle

Legende:

	Aussagenwerte (A-H)
	Wahrheitswerte der Logikgatter (1-6)
	Gesamtergebnis $\hat{=}$ Wahrheitswert des Wurzel-Logikgatters (7)

Die Bedeutung der Buchstaben A-H und Zahlen 1-7 im Kontext der Baumstruktur sind Abbildung 7.1.0.1 zu entnehmen.

Abbildung B.4.0.3: Ausgabe des Wahrheitstabellengenerators für Aussage X (Ausschnitt). Links: Eigene Darstellung; © für Texturen: Mojang; Rechts: © CircuitVerse

C Ergänzendes Material

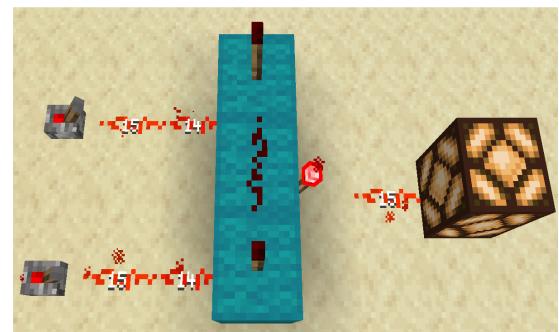
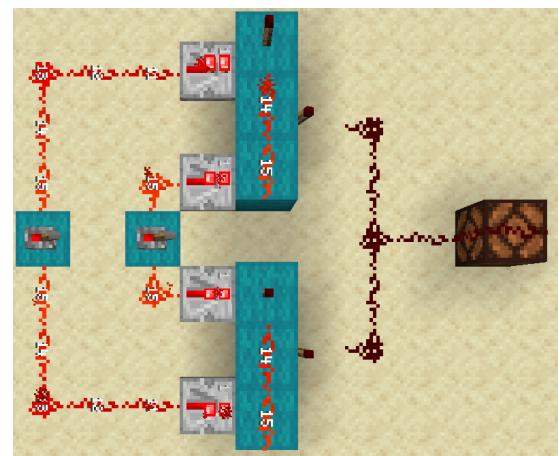
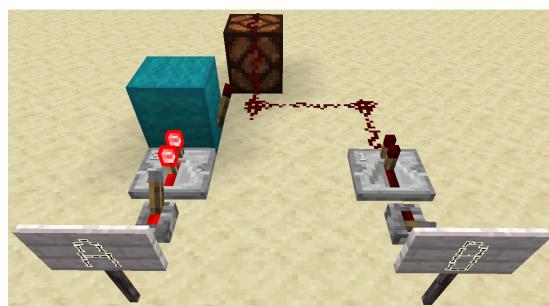
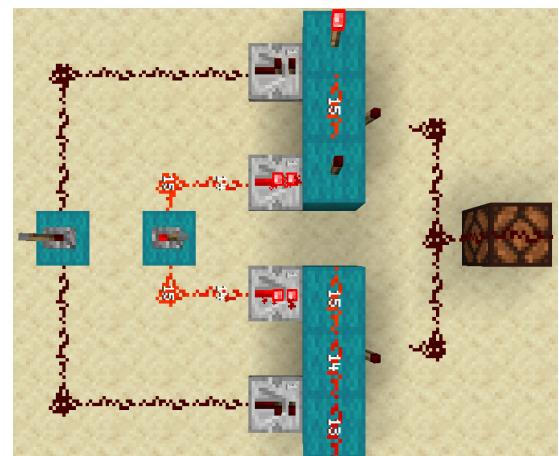
(a) NOT (\neg)(b) AND (\wedge)(c) OR (\vee)(d) XOR (\oplus)(e) Konditional (\rightarrow)(f) Bikonditional (\leftrightarrow)

Abb. C.0.0.1: Umsetzung der Logikoperatoren in Minecraft-Schaltungen. © für Texturen: Mojang Studios

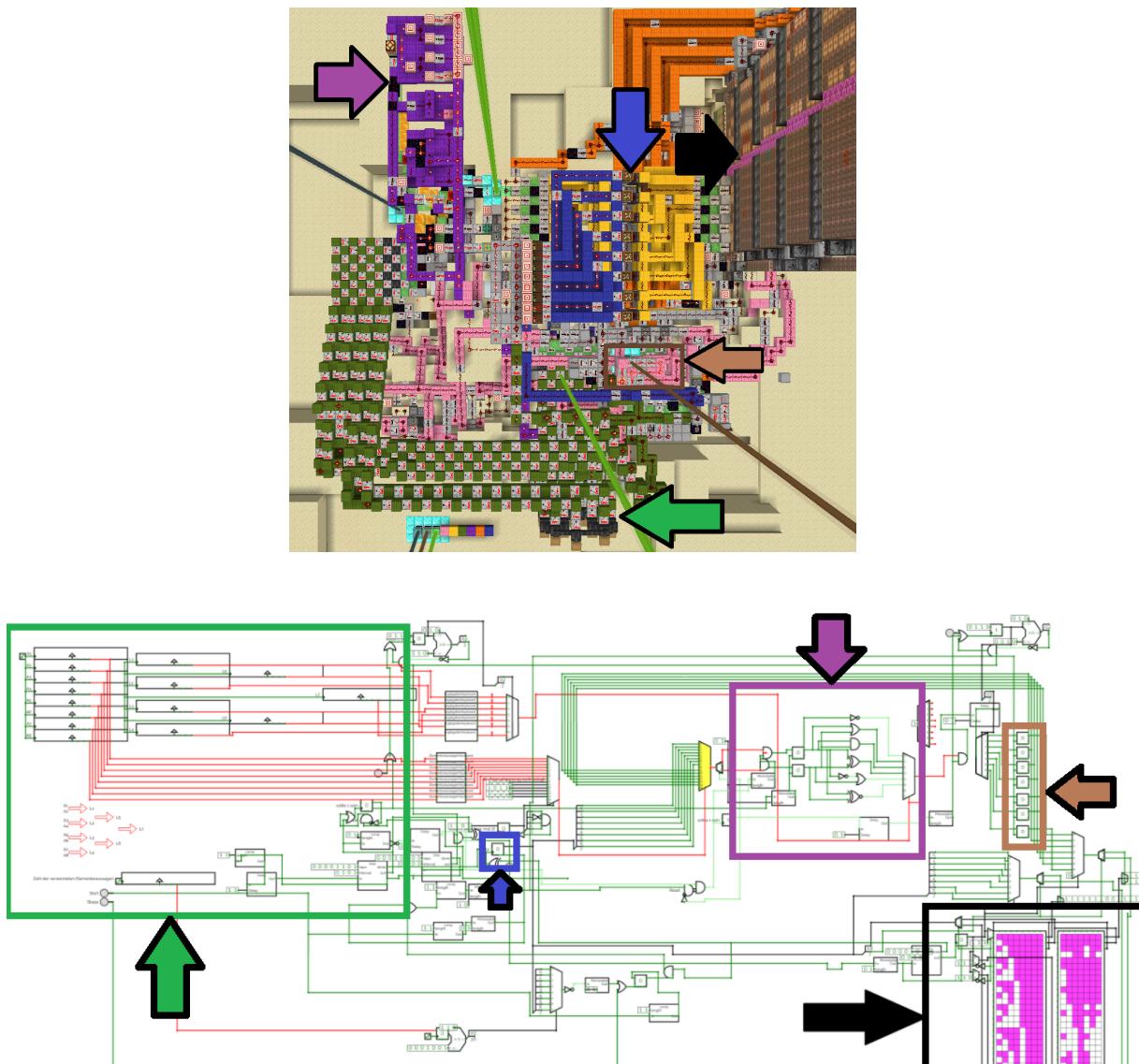


Abbildung C.0.0.2: Übersicht über die beiden Tabellengenerator-Modelle.

Oben: Eigene Darstellung; © für Texturen: Mojang Studios

Unten: Screenshot einer eigens erstellten Schaltung; © CircuitVerse

Legende:

- Position der sechs berechenbaren Logikgatter
- Binärzähler B zum Zählen der Wiederholungen & Festlegung der Wahrheitswerte der Elementaraussagen
- Zwischenspeicher Z für berechnete Wahrheitswerte
- Eingabe der zu überprüfenden Aussage
- Ausgabe der Wahrheitstabelle auf einem Display

D Anlagen

785 D.1 Dokumentation der Nutzung von KI

Chat Nr.	Chatbot (Name, Version, Anbieter, URL)	Verwendungszweck bzw. Einsatzszenario (mit Datum des Einsatzes)	Stellenangabe in der Arbeit (Kapitel- Seiten- und ggf. Zeilenangaben)	ergänzende Hinweise
1	ChatGPT, GPT4, OpenAI, https://openai.com	Generieren eines Binärbaums in Latex-Formatierung (18.4.25)	Abb. 7.1.0.1; S. 20	
2	ChatGPT, GPT4, OpenAI, https://openai.com	Generieren einer Latex-Tabelle der Logikoperatoren (19.4.25) Tab.	Tabelle 1; S. 5	
3	ChatGPT, GPT4, OpenAI, https://openai.com	Erstes Einarbeiten in Latex + Troubleshooting für Kompilierfehler (20.1.25)	-	
4	ChatGPT, GPT4, OpenAI, https://openai.com	Einarbeiten in das Thema Logikgatter (15.1.25)	-	
5	ChatGPT, GPT4, OpenAI, https://openai.com	Erstellen einer Legende für Bilder zu den Flip-Flops inkl. Mini-Symbole (1.6.25)	Abb. 4.1.1.1 / 4.1.2.2; S. 10 / S. 11	
6	ChatGPT, GPT4, OpenAI, https://openai.com	Hilfe zur Erstellung von Funktionsgraphen in Latex (5.11.25)	Abb. 8.2.0.1; S. 27	
7	ChatGPT, GPT4, OpenAI, https://openai.com	Erstellen einer mathematischen Gleichung in Latex Formatierung (6.11.25)	Kap. 8.2 S. 27 Z.533	Vor der Anfrage hatte ich bereits die Idee mit ihm gehabt und durchgerechnet, es ging mir nur um eine schnelle Umsetzung in Latex-Formatierung
8	Gemini, 2.5 Flash, https://gemini.google.com	Hilfe bei Erstellen eines Anhangs in Latex (6.11.25)	-	
9	Gemini, 2.5 Flash, https://gemini.google.com	Hilfe bei der finalen Formatierung (7.11.25)	-	

Tabelle 8: Übersichtstabelle der verwendeten KI-Tools/Chatbots

D.2 Download-Links

Folgende Dateien wurden in einem GitHub-Repository hinterlegt und können unter https://github.com/2-cellfesfsef/W-Seminar_NoahSchuller heruntergeladen werden:

- **Minecraft-Welt-Download:** Der Tabellengenerator wurde in Minecraft-Version 1.21.4 gebaut und getestet. Nach Entpacken der ZIP-Datei kann der Welt-Ordner namens „Wahrheitstabellengenerator“ im Verzeichnis `User/AppData/Roaming/.minecraft/saves` abgelegt und anschließend im Spiel abgerufen werden.
- **CircuitVerse-Projekt:** Die Datei `Wahrheitstabellengenerator.cv` kann nach Öffnen des CircuitVerse-Simulators unter <https://circuitverse.org/simulator> importiert werden. Dazu im Drop-Down-Menü „Project“ die Option „Open Offline“ auswählen und die Datei hochladen.
- **Weitere Dateien:** Außerdem sind die vollständigen Konversationen mit den KI-Chatbots sowie der gesamte Latex-Ordner des Projektes hochgeladen.

Alle Dateien befinden sich außerdem auf dem beigelegten USB-Stick. Bei Fragen und Problemen bitte noah.schuller701152@gmail.com kontaktieren bzw. Kontakt über itslearning aufnehmen.

Literatur

Beckermann, A. (2014). *Einführung in die Logik* (4., durchgesehene Auflage). Walter de Gruyter GmbH Berlin/Boston.

Chip. (2025). *Binärsystem - einfach erklärt*. (Abgerufen am 5. November 2025) https://praxistipps.chip.de/binaersystem-einfach-erklaert_176613

CircuitVerse. (2025a). *Documentation*. (Abgerufen am 6. November 2025) <https://docs.circuitverse.org/>

CircuitVerse. (2025b). *Logic Circuit Simulator*. (Abgerufen am 6. November 2025) <https://circuitverse.org/>

Elektronik-Kompendium. (o. D. a). *Bipolartransistoren*. (Abgerufen am 6. November 2025) <https://www.elektronik-kompendium.de/sites/bau/0201291.htm>

Elektronik-Kompendium. (o. D. b). *Flip Flops*. (Abgerufen am 6. November 2025) <https://www.elektronik-kompendium.de/sites/dig/0209301.htm>

Elektronik-Kompendium. (o. D. c). *JK-Flip-Flop*. (Abgerufen am 6. November 2025) <https://www.elektronik-kompendium.de/sites/dig/0210032.htm>

- Elektronik-Kompendium. (o. D. d). *Monostabile Kippstufe mit Transistoren*. (Abgerufen am 6. November 2025) <https://www.elektronik-kompendium.de/sites/praxis/bausatz-monoflop.htm>
- Elektronik-Kompendium. (2017). *Der analoge Schalter II*. (Abgerufen am 6. November 2025) <https://www.elektronik-kompendium.de/public/schaerer/anasw2.htm>
- Hellmann, R. (2022). *Rechnerarchitektur: Einführung in den Aufbau moderner Computer* (3., korrigierte und erweiterte Auflage). De Gruyter Oldenbourg.
- Minecraft-Wiki. (2025a). *Hauptseite*. (Abgerufen am 6. November 2025) <https://de.minecraft.wiki/>
- Minecraft-Wiki. (2025b). *Redstone-Fackel*. (Abgerufen am 6. November 2025) <https://de.minecraft.wiki/w/Redstone-Fackel>
- Minecraft-Wiki. (2025c). *Redstone-Verstärker*. (Abgerufen am 6. November 2025) <https://de.minecraft.wiki/w/Redstone-Verst%C3%A4rker>
- Properinglish19. (2013, 12. Januar). *Signal Strength Up/Down Counter*. (Abgerufen am 5. November 2025) https://www.youtube.com/watch?v=n4G0Dt_AYR8
- Reichardt, J. (2009). *Lehrbuch Digitaltechnik: Eine Einführung mit VHDL*. Oldenbourg Wissenschaftsverlag.
- Tanenbaum, A. (2010). *Structured Computer Organization* (3rd revised Edition). Pearson Prentice Hall.
- theoryCIRCUIT. (2024). *NOR and NAND Gates using transistor*. (Abgerufen am 8. November 2025) <https://theorycircuit.com/digital-electronics/nor-and-nand-gates-using-transistor/>

Abbildungsverzeichnis

2.1.0.1	Logiksymbole der Boole'schen Algebra. © 2009 Oldenbourg Wissenschaftsverlag GmbH	5
3.1.0.1	Schaltsymbol eines Transistors. Eigene Darstellung.	7
3.2.0.1	Vereinfachte NOR-Schaltung mithilfe von BJTs. Eigene Darstellung.	8
4.1.1.1	Set-Vorgang eines vereinfacht dargestellten S-R-Flip-Flops, bearbeitet. © Engr. Shahzada Fahad	10
4.1.2.2	Toggle-Vorgang eines J-K-Flip-Flops, bearbeitet. © Elektronik-Kompendium	11
4.1.2.1	Vereinfachte Schaltung des D-Flip-Flops. © Elektronik-Kompendium	11
4.2.0.1	3-Bit-Rückwärtszähler. © 2022 Walter de Gruyter GmbH	12

4.3.0.1	Schaltbild eines En- und Decoders. © GeeksforGeeks, Sanchhaya Education Private Limited / Electrical4U	14
4.4.0.1	4:1 Multiplexer. Screenshot einer eigens erstellten Schaltung; © CircuitVerse	14
5.1.0.1	Verknüpfung von Redstone-Leitungen, -Verstärkern und -Fackeln. Eigene Darstellung; © für Texturen: Mojang Studios	16
5.1.0.3	Drei OR-Gatter (links) bzw. NOR-Gatter (rechts), bei denen jeweils 0, 1 bzw. beide Eingänge aktiv sind. Eigene Darstellung; © für Texturen: Mojang Studios	17
5.1.0.2	Clock basierend auf Redstone-Fackeln. Eigene Darstellung; © für Texturen: Mojang Studios	17
7.1.0.1	Binärbaumdarstellung der Aussage X. Eigene Darstellung.	20
7.1.0.2	Multiplexer zur Auswahl der Logikgatter. Screenshot einer eigens erstellten Schaltung; © CircuitVerse	21
7.1.0.3	Multiplexer zur Auswahl der Aussagenwerte. Screenshot einer eigens erstellten Schaltung; © CircuitVerse	21
7.2.0.1	Flussdiagramm zur Berechnung einer Tabellenzeile. Eigene Darstellung.	23
8.2.0.1	Laufzeiten der Modelle in Abhängigkeit der Zahl der verwendeten Elementaussagen n . Eigene Darstellung.	27
A.1.0.1	Kompaktes AND-Gatter, bei dem aktuell beide Ausgänge aktiviert sind. Eigene Darstellung; © für Texturen: Mojang Studios	30
A.1.0.2	Redstone-Signalstärke-Decoder. Eigene Darstellung; © für Texturen: Mojang Studios	31
A.1.0.3	Erweiterung eines Decoders zum Multiplexer (Seitenansicht). Eigene Darstellung; © für Texturen: Mojang Studios	31
A.1.0.4	Monostabile Kippstufe nach Aktivierung. Eigene Darstellung; © für Texturen: Mojang Studios	32
A.1.0.5	J-K-Flip-Flop basierend auf Spendern. Eigene Darstellung; © für Texturen: Mojang Studios	32
A.1.0.6	Signalstärke-Zähler, abgewandeltes Design nach Properinglish19, 2013. Eigene Darstellung; © für Texturen: Mojang Studios	33
A.2.0.1	Verzögerungs-Sub-Circuit. Screenshot einer eigens erstellten Schaltung; © CircuitVerse	34
A.2.0.2	Monostabile Kippstufe als Sub-Circuit. Screenshot einer eigens erstellten Schaltung; © CircuitVerse	34

A.2.0.3	Schaltkreis zur Signalverlängerung. Screenshot einer eigens erstellten Schaltung; © CircuitVerse	35
A.2.0.4	Sub-Circuit für Schleifen. Screenshot einer eigens erstellten Schaltung; © CircuitVerse	36
B.1.0.1	Binärzähler B1 des Wahrheitstabellengenerators. Eigene Darstellung; © für Texturen: Mojang Studios	37
B.2.0.1	Clock zur Koordination der Abläufe. Eigene Darstellung; © für Texturen: Mojang Studios	38
B.4.0.1	Eingabe des Wahrheitstabellengenerators. Eigene Darstellung; © für Texturen: Mojang Studios	39
B.4.0.2	Benutzerschnittstelle in CircuitVerse. Screenshot einer eigens erstellten Schaltung; © CircuitVerse	40
B.4.0.3	Ausgabe des Wahrheitstabellengenerators für Aussage X (Ausschnitt). Links: Eigene Darstellung; © für Texturen: Mojang; Rechts: © CircuitVerse	41
C.0.0.1	Umsetzung der Logikoperatoren in Minecraft-Schaltungen. © für Texturen: Mojang Studios	42
C.0.0.2	Übersicht über die beiden Tabellengenerator-Modelle. Oben: Eigene Darstellung; © für Texturen: Mojang Studios; Unten: © CircuitVerse	43

Tabellenverzeichnis

1	Logische Operatoren und ihre Bedeutung	5
2	Wahrheitstabelle für die Aussage $(A \wedge B) \rightarrow C$	7
3	Wahrheitstabelle des S-R-Flip-Flops	9
4	Wahrheitstabelle des T-Flip-Flops	11
5	Ausgegebene Binärzahlen eines 3-Bit-Rückwärts- und Vorwärtszählers nach m Taktimpulsen	12
6	Logikoperatoren und ihre Umformungen basierend auf den Operatoren NOT (\neg) und OR (\vee)	18
7	Berechnungszeiten und Verhältnisse zwischen CircuitVerse (links) und Minecraft (rechts)	25
8	Übersichtstabelle der verwendeten KI-Tools/Chatbots	44

Persönliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Seminararbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die im Literaturverzeichnis angeführten Quellen und Hilfsmittel verwendet habe.

Insbesondere versichere ich, dass alle Stellen der Seminararbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen sind, durch Angabe der Herkunft kenntlich gemacht wurden. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie Quellen aus dem Internet.

Die Seminararbeit ist in dieser oder einer ähnlichen Form in keinem anderen Kurs des diesjährigen oder eines vorhergehenden Abiturjahrgangs vorgelegt worden.

Erweiterte Schlusserklärung:

Bei der Anfertigung der vorliegenden Arbeit habe ich KI-Tools/Chatbots (vgl. beiliegende Übersichtsliste) verwendet. Nach dem Einsatz habe ich die generierten Ergebnisse jeweils vollumfänglich geprüft und überarbeitet. Ich versichere die lückenlose Einhaltung der erlernten wissenschaftlichen Standards und übernehme die volle Verantwortung für die gesamte vorliegende Arbeit. Außerdem wurde die Formatierung in Latex durch KI unterstützt.

Mir ist bekannt, dass die vorliegende Arbeit mit 0 Punkten bewertet werden kann, wenn gegen diese Grundsätze verstoßen wurde, und dass in diesem Fall eine Zulassung zum Abitur nicht möglich ist.

Nürnberg, den _____

(Datum)

_____ (Unterschrift der Schülerin / des Schülers)