Friedrich-Schiller Universität Jena

Fakultät für Mathematik und Informatik

Chemisches Digitalcomputermodell eines endlichen Automaten, der ohne Rest durch drei teilbare nat. Zahlen erkennt

Projektarbeit zum Modul molekulare Algorithmen

Tom Prantz Matrikel 168557

Celestino Madera Castro Matrikel 153138

betreut von

Dr. Thomas Hinze

Sommersemester 2020

Inhaltsverzeichnis

1. Einleitung

Die Bioinformatik ist eine interdisziplinäre Wissenschaft, welche die Probleme der Biowissenschaften mit computergestützten Methoden löst. Bereits in der Vergangenheit trug sie zu grundlegenen Erkenntnissen der modernen Biologie und Medizin bei. So leistete die Bioinformatik beispielsweise im Jahre 2001 einen wesentlichen Beitrag zur Sequenzierung des menschlichen Genoms.

Aufgrund der Vielzahl an biologischen Herausforderungen, welche durch informatische Werkzeuge und Hilfsmittel bewältigt werden können, entstanden im Laufe der Zeit verschiedene Fachbereiche der Bioinformatik. Eines dieser verschiedenen Fachgebiete ist beispielsweise die molekulare Bioinformatik, wozu beispielsweise Themen wie DNA-Computing oder Sequenzanalyse gehören.

Das Thema molekulare Algorithmen zählt ebenfalls zu diesem Fachgebiet. Mittels molekularer Algorithmen sollen die technischen Prinzipien von Computern in einem chemischen Modell dargestellt werden, da biologische System der konventionellen Rechentechnik in vielen Aspekten überlegen ist. Beispielsweise kann durch die Verwendung von DNA als Speichermedium eine deutlich höhere Speicherdichte und -persistenz erreicht werden. Ebenfalls kann durch hocheffiziente, chemische Verarbeitung ein deutlich niedrigerer Energieverbrauch erzielt werden.

Aufgrund dieser Vorteile ist das Ziel dieser Hausarbeit, einen deterministischen, endlichen Automaten in einem chemischen Digitalcomputermodell auf Grundlage molekularer Algorithmen zu implementieren und zu simulieren.

2. Aufgabenstellung

Das Ziel dieser Hausarbeit ist es, ein chemisches Digitalcomputermodell eines deterministischen, endlichen Automaten zu erstellen, welcher ohne Rest durch drei teilbare natürliche Zahlen akzeptiert.

Der gesuchte deterministische, endliche Automat soll ziffernweise eine natürliche Dezimalzahl einlesen, welche aus beliebig vielen Dezimalziffern besteht. Er soll genau die Dezimalzahlen akzeptieren, die ohne Rest durch 3 teilbar sind ($L(M) = \{0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, ...\}$).

Dieser Automat soll mit 3 Zuständen auskommen und so modifiziert werden, dass er unter Verwendung einer binären Kodierung in ein chemisches Digitalcomputermodell überführt werden kann. Dabei sollen alle Zustände des Automaten binär dargestellt werden. Weiterhin soll jedes Eingabezeichen wiederum durch 4 Bits repräsentiert werden.

Für diesen Automaten sollen Schaltfunktionen ermittelt werden, welche gegebenenfalls mittels dem Karnaugh-Plan oder einer anderen Technik optimiert werden. Mithilfe dieser Schaltfunktionen wird wiederum ein Reaktionsnetzwerk aufgestellt, für das sinnvolle Reaktionsparameter spezifiziert werden sollen. Anschließend soll das Netzwerkverhalten für mehrere Fallstudien simuliert werden.

3. Herleitung und Definition des Automaten

Um die Aufgabenstellung zu bewältigen, wird ein herkömmlicher, deterministischerer, endlicher Automat definiert, welcher Dezimalzahlen ziffernweise einlesen kann. Dieses Modell reicht jedoch nicht aus, um einen solchen Automaten in einem chemischen Digitalcomputermodell zu implementieren. Deshalb sollte der spezifizierte Automat in einer Binärdarstellung repräsentiert werden. Das hat zur Folge, dass die einzelnen, binär kodierten Ziffern einer Dezimalzahl nun bit-weise eingelesen werden, wodurch das Aufstellen von Schaltfunktionen für einen chemischen Digitalcomputer vereinfacht wird.

3.1. Definition eines deterministischen, endlichen Automaten

Um einen deterministischen, endlichen Automaten (kurz: *DEA*) zur Lösung der Herausforderzung zu erstellen, werden zuerst die wichtigsten Eigenschaften eines solchen Automaten charakterisiert.

Ein DEA ist immer eindeutig. Das heißt, dass für jeden Zustand eines solchen Automaten in Abhängigkeit von der Eingabe der Übergang in einen anderen Zustand (oder den gleichen) eineindeutig definiert ist. Darüber hinaus werden für jeden DEA eine bestimmte Menge an Start- und Endzuständen definiert. Ein eingegebenes Wort wird nur dann vom DEA akzeptiert, wenn dieser deterministische, endliche Automat in einem Finalzustand endet.

Um einen für die Aufgabenstellung benötigten Automaten zu definieren, werden die Eigenschaften von durch 3 teilbaren, natürlichen Zahlen sowie von Modulo-Operationen verwendet. Die Quersumme einer jeden durch 3 teilbaren, natürlichen Zahl ist ebenfalls durch 3 teilbar. Dementsprechend kann durch Addition der einzelnen Ziffern einer Dezimalzahl einfacher ermittelt werden, ob die entsprechende Zahl durch 3 teilbar ist oder nicht.

Summiert man diese Zahlen modulo 3 miteinander auf, lässt sich der Aufwand sogar auf ein Minimum beschränken. Das liegt daran, dass nur noch mit den Restklassen 0, 1 und 2 gerechnet werden muss, da jede eingegebene Ziffer eineindeutig einer Restklasse zugewiesen werden kann. Dabei macht die genaue Größe einer Zahl

für Additionen gleicher moduli keinen Unterschied, da nur der Rest einer Zahl das Ergebnis einer Modulo-Operation beeinflusst. Somit ist eine eingegebene Zahl nur durch 3 teilbar, falls das Endergebnis aller Modulo-3-Additionen der Restklasse 0 entspricht.

Eine mögliche grafische Darstellung eines solchen DEA kann Abbildung 1 entnommen werden. Die formale Beschreibung des Automaten (M) sieht hingegen wie folgt aus:

- $M = DEA(Q, \Sigma, \delta, q_0, F)$
- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a_0, ..., a_9\}$ mit $a_i = i$
- $F = q_0$

Dabei stellt Q die endliche Zustandsmenge dar. Es enthält die Zustände q_0, q_1 und q_2, welche die Restklassen $0, 1$ und 2 mod 3 repräsentieren.

Σ bezeichnet wiederum das endliche Eingabealphabet, welches der Menge der erlaubten Eingabesymbole entspricht. Die Eingabesymbole werden für diesen Automaten mit a_i bezeichnet und entsprechen den Ziffern von 0 bis 9 nach folgender mathematischer Regel: $a_i = i$.

δ stellt die Zuständsübergangsfunktion dar, welche wie folgt definiert ist: $\delta\colon Q \times \Sigma \to Q$. Mittels dieser Funktion wird jedem Paar bestehend aus einem Zustand $q \in Q$ und einem Eingabesymbol $a \in \Sigma$ ein Folgezustand $p \in Q$ zugeordnet. Weiterhin wird $q_0 \in Q$ als einziger Startzustand definiert. F enthält wiederum die Menge aller Finalzustände, welcher für diesen Automaten ebenfalls nur q_0 entspricht.

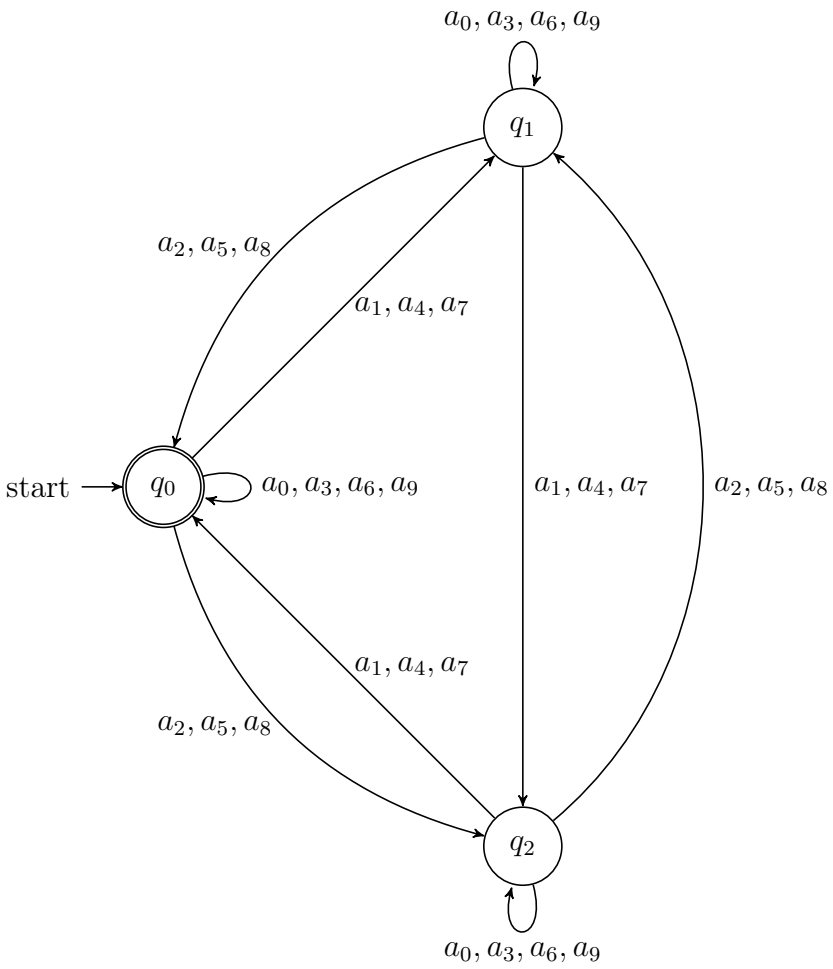


Abbildung 1: Graphendarstellung des DEA mit dezimaler Eingabe

3.2. Herleitung eines Automaten für ein chemisches Digitalcomputermodell

Der aufgestellte, deterministische, endliche Automat akzeptiert alle natürlichen Dezimalzahlen, welche durch drei teilbar sind. Dabei werden die einzelnen Ziffern schrittweise als Dezimalzahlen eingelesen. Da dieser Automat jedoch in einem chemischen Digitalcomputermodell implementiert werden soll, bietet es sich an, alle Komponenten des Automaten in binärer Form darzustellen.

Um den Automaten in einer Binärdarstellung zu repräsentieren, muss das Eingabealphabet auf die Zeichen 0 und 1 reduziert werden. Das hat zur Folge, dass die ursprünglich einzulesenden Zeichen nun anders dargestellt werden müssen. Dazu werden diese Zeichen binär kodiert, indem jedes dieser 10 Zeichen eine eindeutige

4-stellige Bitfolge zugewiesen wird. Dadurch setzt sich jedes ursprüngliche Zeichen aus 4 binären Zeichen zusammen, welche von rechts nach links einzeln eingelesen werden.

Die ursprünglichen Zustandsübergänge erfolgen in diesem Modell stets nachdem 4 binäre Zeichen eingegeben wurden. Dementsprechend benötigt der modifizierte Automat weitere Zustände in Form von Hilfszuständen, um die Zustandsübergänge zwischen zwei binären Zeichen zu simulieren. Zusätzlich sollte die Zustandsübergangsfunktion entsprechend an das neue Modell angepasst werden.

Der modifizierte Automat $M = DEA(Q, \Sigma, \delta, q_0, F)$ (s. Abbildung 2) wird demnach wie folgt definiert:

- $\Sigma = (0, 1)$ mit (s. Tabelle 11)
  - $a_k = 8 * b_1 + 4 * b_2 + 2 * b_3 + b_4$
  - und $b_1, b_2, b_3, b_4 \in \Sigma$
- $Q = (q_0, q_1, ..., q_{11})$ mit der Eigenschaft (s. Tabelle 10)
  - $\forall q_i \in Q : q_i = q_{(z_0,z_1,z_2,z_3)} = q_(8 * z_0 + 4 * z_1 + 2 * z_2 + z_3)$
  - für $i = \{0, 1, ..., 11\}$, $z_k = \{0, 1\}$ und $k = \{0, 1, 2, 4\}$
- $F = q_0 = q_{(z_0=0,z_1=0,z_2=0,z_3=0)}$
- Zustandsübergangsfunktion δ (s. Tabelle 12)

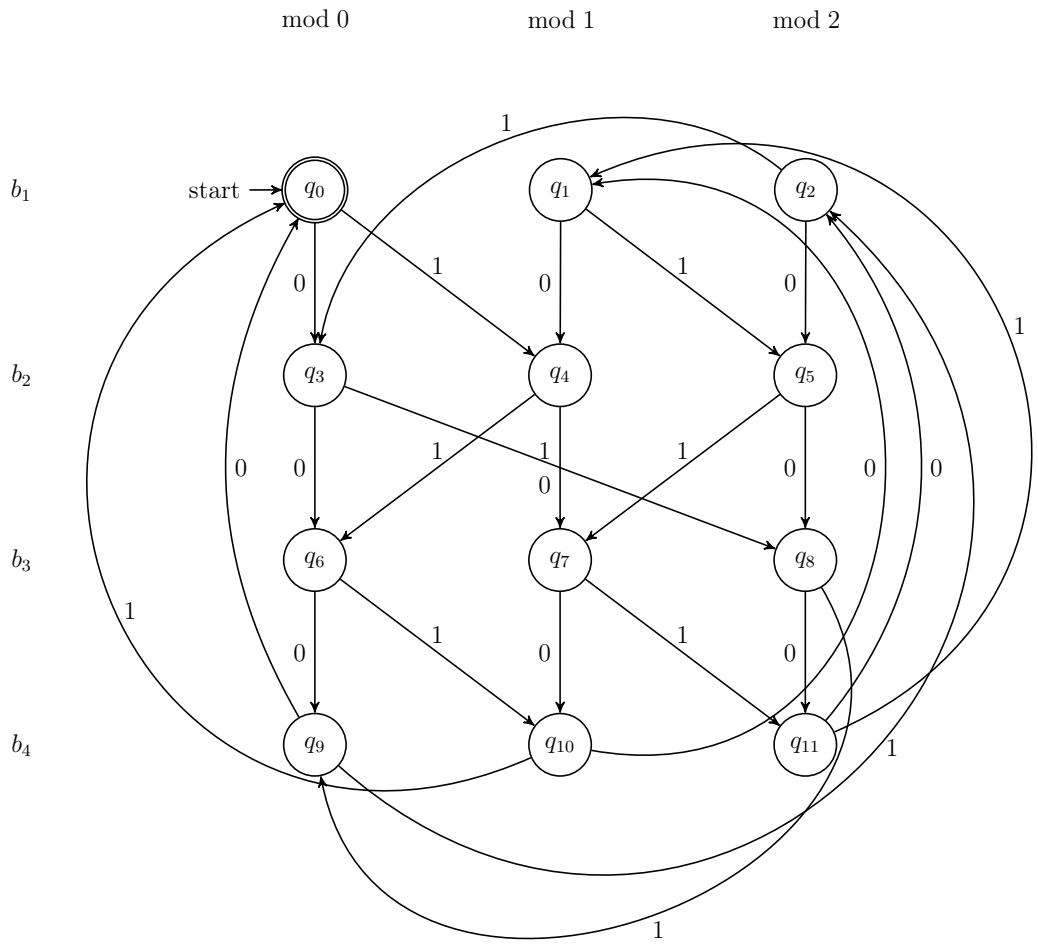


Abbildung 2: Graphendarstellung des DEA mit binäer Eingabe

4. Ermittlung der Schaltung

Mithilfe der Zustandsübergangstabelle (s. Tabelle 12) des binär kodierten DEA können die im Reaktionsnetzwerk zu implementierenden Schaltfunktionen ermittelt werden. Um diese booleschen Schaltfunktionen möglichst simpel zu formulieren, werden Möglichkeiten zur Reduzierung ihrer Komplexität verwendet. Aus diesen optimierten Funktionen werden letztendlich logische Schaltungen ermittelt, die in ein chemisches Digitalcomputermodell implementiert werden können.

4.1. Ermittlung der booleschen Schaltfunktion

Wie bereits erwähnt, werden anhand der Zustandsübergangstabelle die einzelnen, booleschen Schaltfunktionen zur Simulation der Zustandsübergänge formuliert.

Dazu werden zuerst für jedes einzelne Zielzustands-Bit z_i' die jeweiligen Kombinationen der Eingabe-Bits ermittelt. Dabei setzen sich die Eingabe-Bits aus den 4 Bits des Ausgangszustandes (z_0, z_1, z_2, z_3) sowie aus dem aktuell gelesenen Bit der Eingabe (b) zusammen. Diese einzelnen Eingabe-Bits werden als Min-Terme zur Repräsentation einer jeden Kombination durch einen logischen UND-Operanden ($\wedge$) miteinander verknüpft. Die einzelnen Min-Terme eines jeden Zielzustands-Bits werden wiederum mittels eines logischen ODER-Operanden ($\vee$) zusammengefasst. Dadurch erhält man für jedes Zielzustands-Bit z_i' eine boolesche Schaltfunktion in *disjunktiver Normalform* (kurz: *DNF*).

Die aus der Zustandsübergangstabelle resultierenden Schaltfunktionen für z_0', z_1', z_2' und z_3' können Tabelle 1 entnommen werden.

z'_i	z'_0	z'_1	z'_2	z'_3
boolesche	$(\bar{z}_0\bar{z}_1z_2z_3b) \vee$	$(\bar{z}_0\bar{z}_1\bar{z}_2z_3\bar{b}) \vee$	$(\bar{z}_0\bar{z}_1\bar{z}_2\bar{z}_3\bar{b}) \vee$	$(\bar{z}_0\bar{z}_1\bar{z}_2\bar{z}_3\bar{b}) \vee$
Schaltfunktion	$(\bar{z}_0z_1\bar{z}_2z_3\bar{b}) \vee$	$(\bar{z}_0\bar{z}_1\bar{z}_2\bar{z}_3b) \vee$	$(\bar{z}_0\bar{z}_1z_2\bar{z}_3b) \vee$	$(\bar{z}_0\bar{z}_1\bar{z}_2z_3b) \vee$
	$(\bar{z}_0z_1z_2\bar{z}_3\bar{b}) \vee$	$(\bar{z}_0\bar{z}_1z_2z_3b) \vee$	$(\bar{z}_0\bar{z}_1z_2z_3\bar{b}) \vee$	$(\bar{z}_0\bar{z}_1z_2\bar{z}_3\bar{b}) \vee$
	$(\bar{z}_0z_1z_2\bar{z}_3b) \vee$	$(\bar{z}_0\bar{z}_1z_2\bar{z}_3\bar{b}) \vee$	$(\bar{z}_0z_1\bar{z}_2\bar{z}_3\bar{b}) \vee$	$(\bar{z}_0z_1\bar{z}_2\bar{z}_3\bar{b}) \vee$
	$(\bar{z}_0z_1z_2z_3\bar{b}) \vee$	$(\bar{z}_0\bar{z}_1z_2z_3\bar{b}) \vee$	$(\bar{z}_0z_1\bar{z}_2\bar{z}_3b) \vee$	$(\bar{z}_0z_1\bar{z}_2z_3b) \vee$
	$(\bar{z}_0z_1z_2z_3b) \vee$	$(\bar{z}_0z_1\bar{z}_2\bar{z}_3\bar{b}) \vee$	$(\bar{z}_0z_1\bar{z}_2z_3b) \vee$	$(\bar{z}_0z_1\bar{z}_2\bar{z}_3\bar{b}) \vee$
	$(z_0\bar{z}_1\bar{z}_2\bar{z}_3\bar{b}) \vee$	$(\bar{z}_0z_1\bar{z}_2\bar{z}_3b) \vee$	$(\bar{z}_0z_1z_2\bar{z}_3b) \vee$	$(\bar{z}_0z_1z_2\bar{z}_3\bar{b}) \vee$
	$(z_0\bar{z}_1\bar{z}_2\bar{z}_3b)$	$(\bar{z}_0z_1\bar{z}_2z_3b)$	$(\bar{z}_0z_1z_2z_3\bar{b}) \vee$	$(\bar{z}_0z_1z_2z_3b) \vee$
			$(\bar{z}_0z_1z_2z_3b) \vee$	$(z_0\bar{z}_1\bar{z}_2\bar{z}_3\bar{b}) \vee$
			$(z_0\bar{z}_1\bar{z}_2\bar{z}_3\bar{b}) \vee$	$(z_0\bar{z}_1\bar{z}_2\bar{z}_3b) \vee$
			$(z_0\bar{z}_1\bar{z}_2z_3b) \vee$	$(z_0\bar{z}_1z_2\bar{z}_3\bar{b}) \vee$
			$(z_0\bar{z}_1z_2z_3\bar{b})$	$(z_0\bar{z}_1z_2z_3b)$

Tabelle 1: Boolesche Schaltfunktionen

4.2. Optimierung der Schaltfunktion mittels Karnaugh-Plan

Für jedes Zielzustands-Bit z_i wurde eine boolesche Schaltfunktion in disjunktiver Normalform ermittelt (s. Tabelle 1). Da die Komplexität der Schaltfunktionen sehr hoch ist, sollten diese Funktionen vereinfacht werden, indem sie in ihre *orthogonale disjunktive Normalform* überführt werden.

Dazu kann beispielsweise der Karnaugh-Plan verwendet werden, welcher es ermöglicht aus jeder beliebigen disjunktiven Normalform, einer booleschen Schaltfunktion, einen minimalen disjunktiven logischen Ausdruck zu erzeugen. Dabei werden die einzelne Konjunktionen paarweise zusammengefasst, sobald sich in beiden Teilausdrücken nur eine Disjunktion unterscheidet. Diese Disjunktion entfällt wiederum in dem zusammengefassten Minterm, aufgrund folgender boolescher Eigenschaften:

- $a \vee \bar{a} = 1$
- $(a \wedge b \wedge c) \vee (a \wedge b \wedge \bar{c}) = a \wedge b \wedge (c \vee \bar{c}) = a \wedge b \wedge 1 = a \wedge b$

Auf Grundlage dieser Regel werden alle Minterme solange rekursiv zusammengefasst, bis sich alle Minterme in mehr als nur einer Disjunktion unterscheiden. Die mittels Karnaugh-Plan optimierten, booleschen Schaltfunktionen für z'_0, z'_1, z'_2 und z'_3 können Tabelle 2 entnommen werden.

z'_i	z'_0	z'_1	z'_2	z'_3
boolesche	$(z_0\bar{z_1}\bar{z_2}\bar{z_3}) \vee$	$(\bar{z_0}z_1\bar{z_2}\bar{z_3}) \vee$	$(z_0\bar{z_1}\bar{z_2}z_3b) \vee$	$(z_0\bar{z_1}z_2z_3b) \vee$
Schaltfunktion	$(\bar{z_0}z_1z_2) \vee$	$(\bar{z_0}\bar{z_1}z_2\bar{b}) \vee$	$(\bar{z_0}z_1z_2z_3) \vee$	$(z_0\bar{z_1}\bar{z_2}\bar{z_3}) \vee$
	$(\bar{z_0}z_1z_3\bar{b}) \vee$	$(\bar{z_0}\bar{z_1}z_3\bar{b}) \vee$	$(\bar{z_0}z_1\bar{z_2}\bar{z_3}) \vee$	$(\bar{z_0}z_1z_3b) \vee$
	$(\bar{z_0}z_2z_3b)$	$(\bar{z_0}\bar{z_2}b)$	$(\bar{z_0}z_1b) \vee$	$(\bar{z_0}\bar{z_1}z_2\bar{z_3}) \vee$
			$(\bar{z_0}z_2\bar{z_3}b) \vee$	$(\bar{z_0}\bar{z_2}z_3b) \vee$
			$(\bar{z_1}z_2z_3\bar{b}) \vee$	$(\bar{z_0}\bar{z_3}\bar{b}) \vee$
			$(\bar{z_1}\bar{z_2}\bar{z_3}\bar{b})$	$(\bar{z_1}\bar{z_3}\bar{b})$

Tabelle 2: Optimierte boolesche Schaltfunktionen

4.3. Herstellung der logischen Schaltung für das chemische Digitalcomputermodell

Um die optimierte, boolesche Schaltfunktion in ein chemisches Digitalcomputermodell zu implementieren, müssen diese Funktionen als logische Schaltungen abgebildet werden. Die einzelnen Eingabe-Bits werden innerhalb der logischen Schaltungen als Signale dargestellt. Zur Abbildung der einzelnen Konjunktionen werden die entsprechenden Signale durch ein einziges logisches UND-Gatter miteinander verbunden. Dabei werden alle innerhalb der jeweiligen Konjunktion negierten Signale am Eingang des UND-Gatters ebenfalls negiert. Die für eine Schaltfunktion erstellten UND-Gatter werden wiederum durch ein logisches ODER-Gatter miteinander verknüpft.

Die logische Schaltung für z_0 ist der Abbildung 3 zu entnehmen. Die Schaltungen für z_1, z_2 und z_3 werden in den Abbildungen 4, 5 und 6 dargestellt.

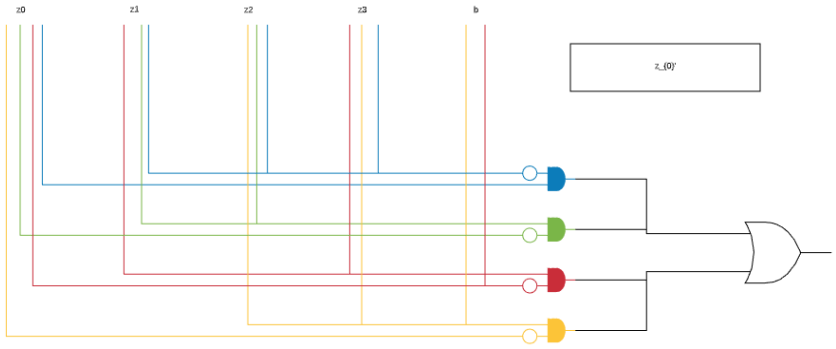


Abbildung 3: Logische Schaltung für z_0

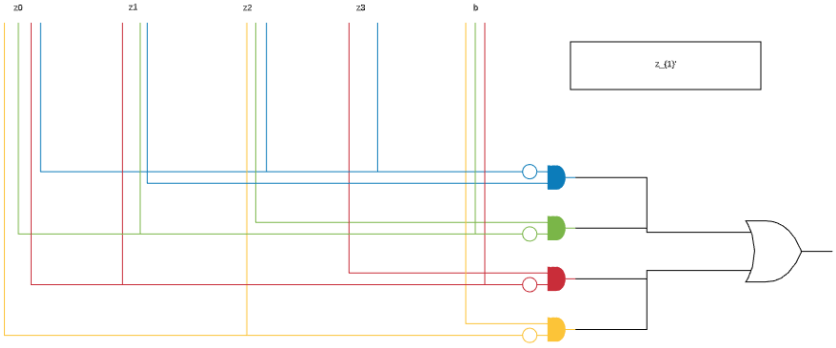


Abbildung 4: Logische Schaltung für z_1

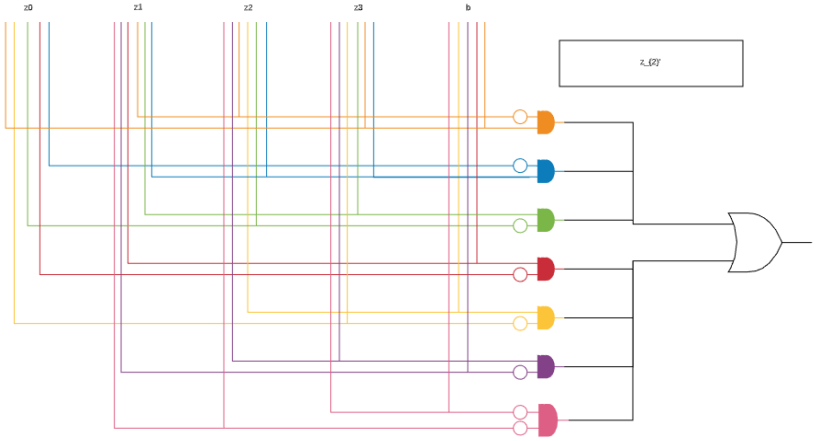


Abbildung 5: Logische Schaltung für z_2

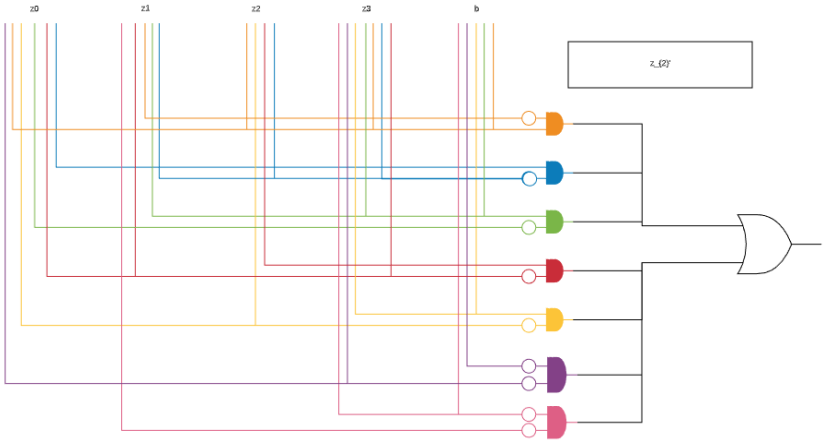


Abbildung 6: Logische Schaltung für z_3

5. Aufstellung des Reaktionsnetzwerks

Auf Grundlage der in Kapitel 4.3 erstellten Schaltungen, kann ein Reaktionsnetzwerk für einen chemischen Digitalcomputermodell erstellt werden. Dazu müssen Reaktionen für die in den logischen Schaltungen enthaltenen Gatter definiert werden, welche das jeweilige Gatter im chemischen Digitalcomputermodell abbilden. Darüber hinaus müssen für dieses Modell verschiedene Spezien definiert werden, welche Eingangs- und Ausgangssignale der Schaltungen repräsentieren sowie Zwischenergebnisse diverser logischer Operationen darstellen.

Zur korrekten Verarbeitung des eingegebenen Wortes muss dem Modell ein Taktsignal hinzugefügt werden. Mittels dieses Signals soll das zu überprüfende Wort bit-weise eingelesen werden. Dabei sollte dieses Signal ebenfalls auf einem Reaktionsnetzwerk basieren.

5.1. Definition eines Brusselators zur Erzeugung spikeförmiger Oszillationen

Um die Funktionalität des deterministischen, endlichen Automaten im chemischen Digitalcomputermodell zu gewährleisten, wird ein Taktsignal benötigt. Mithilfe dieses Taktsignals soll das Eingabewort bit-weise eingelesen und vom Automaten entsprechend verarbeitet werden. Dabei sollte dieses Triggersignal möglichst rechteckig verlaufen, um die Funktionalität eines binären Schalters zu simulieren.

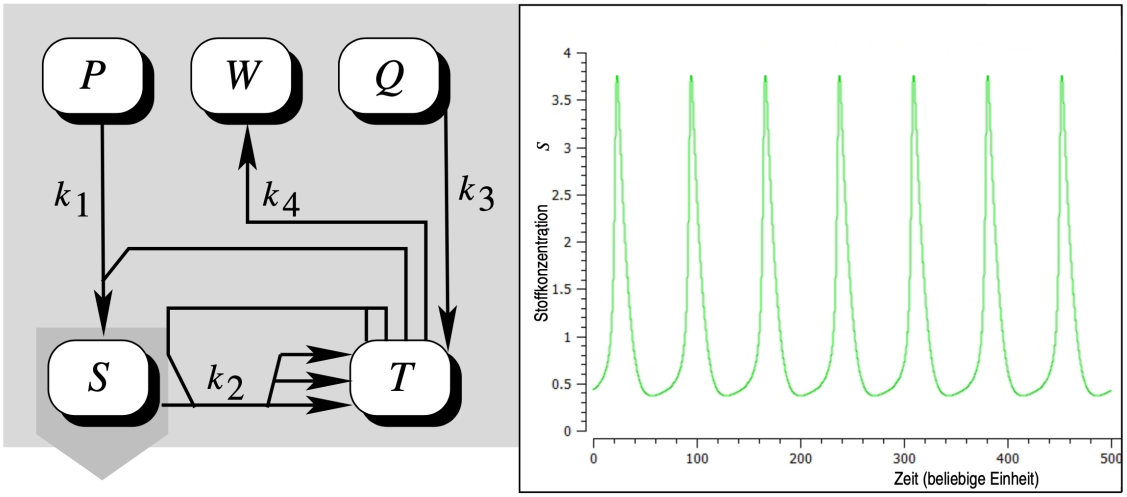
Um ein solches rechteckiges Triggersignal zu erzeugen, kann beispielsweise ein Brusselator in Kombination mit einem binären Signalseperator verwendet werden. Ein Brusselator ist ein einfaches, anorganisches Reaktionssystem, welches zu einem dauerhaft oszillierenden Verhalten fähig ist. Mithilfe eines solchen Brusselators können spikeförmige Oszillationsverläufe erzeugt werden. Diese ungedämpfte Oszillation entsteht durch eine positive Rückkopplung im Reaktionssystem. Dabei wird derselbe Stoff in einer Reaktion konsumiert sowie gebildet. Dieser Prozess wird als Autokatalyse bezeichnet, dessen Effekt verstärkt werden kann, indem noch eine reaktionsübergreifende positive Rückkopplung über die Spezies T durchgeführt wird. Um eine oszillationsfähige Rückkopplung hervorzurufen, müssen stöchiometrische

Faktoren ≥ 3 einbezogen werden. Ein Brüsselator kann beispielsweise durch das folgende Differentialgleichungssystem dargestellt werden:

- $[\dot{P}] = -k1[P][T]$
- $[\dot{Q}] = -k3[Q]$
- $[\dot{S}] = k1[P][T] - k2[S][T]^2$
- $[\dot{T}] = -k1[P][T] + k2[S][T]^2 + k3[Q] - k4[T]$
- $[\dot{W}] = k4[T]$

Die Frequenz der Oszillation wird in diesem Modell durch die Geschwindigkeit der Zerfallsreaktion T $\xrightarrow{k_4}$ W bestimmt. In Folge dieser Reaktion ensteht das Abfallprodukt W, welches schubweise produziert wird und sich zunehmend im Reaktionssystem ansammelt.

Die spikeförmige Oszillation zeichnet sich hingegen durch ein niedriges Stoffkonzentrationsniveau aus, welches sich mit kurzzeitigen, heftigen Impulsen (Spikes) abwechselt (s. Abbildung 7). Diese Impulse führen zu starken Konzentrationsänderungen, welche schnell ansteigen und daraufhin ebenfalls unmittelbar schnell wieder abfallen. Auf Grundlage dieses Verhaltens können nachgeschaltete Reaktionen die Intensität dieser wiederkehrenden Impulse leicht erkennen und somit als Triggersignal verwenden.



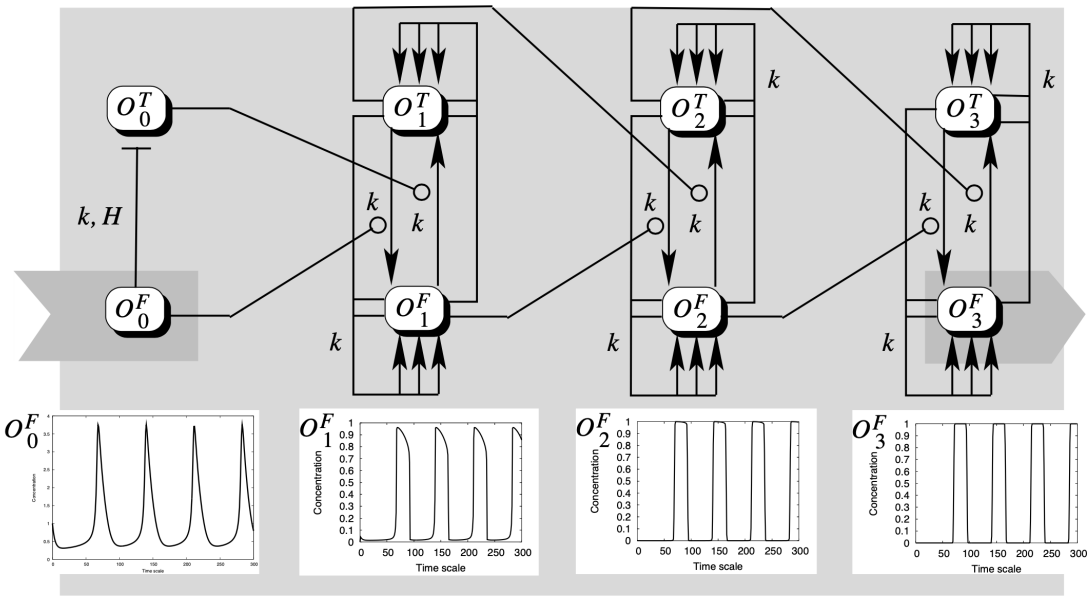
Quelle: „Computer der Natur“

Abbildung 7: Reaktionsnetzwerk des Brusselators, bestehend aus vier Einzelreaktionen: (1) P + T $\xrightarrow{k_1}$ S; S + 2T $\xrightarrow{k_2}$ 3T; (3) Q $\xrightarrow{k_3}$ T; (4) T $\xrightarrow{k_4}$ W. Rechts: Konzentrations-Zeit-Verlauf $[S](t)$ resultiert aus der Parametrisierung $k_1 = k_2 = k_3 = k_4 = 0,\ 1$ und den Anfangskonzentrationen $[P](0) = 3$; $[Q](0) = 0$; $[W](0) = 0,\ 5$; $[T](0) = 0,\ 5$.

5.2. Konvertierung des Oszillationsverlaufs mithilfe eines binären Signalseparators

Zur Simulation eines binären Schalters erweist es sich in der Regel als vorteilhaft, klar interpretierbare und schnell umschaltende Signalverläufe als Triggerfunktion zu verwenden. Dabei sollte ein gezielter Signalsprung pro Schaltvorgang stattfinden. Dementsprechend eignen sich am besten Oszillationsverläufe, deren Wellenform der Ausgabe eines Taktgenerators nahekommt und idealerweise aus fortlaufenden Rechteckimpulsen besteht.

Mithilfe eines binären Signalseparators kann der spikeförmige Oszillationsverlauf eines Brusselators in angenäherte Rechteckimpulse umgewandelt werden. Dabei wird ein solcher Signalseparator zwischen dem Oszillatorausgang des Brusselators und den zu triggernden Reaktionsketten implementiert und als mehrstufige Reaktionskaskade ausgeführt (s. Abbildung 8).



Quelle: „Computer der Natur“

Abbildung 8: Reaktionskaskade zur binären Signalseparation des Eingabeoszillationsverlaufs $[O_0^F]$. Konzentrationswerte ≥ 1 und solche, die nahe bei 1 liegen, werden asymptotisch zu 1 konvertiert, während Werte unterhalb des Schwellwertes H gegen 0 gedrückt werden. Michaelis-Menten-Kinetik in Kombination mit Massenwirkungskinetik beschreibt das dynamische Verhalten des Signalseparators. Ausgehend von der Ausgabe eines vorgeschalteten Brusselators zeigt die Simulationsstudie die Signalseparation unter Benutzung der Parametrisierung: $H = 0,6;\ k = 0,1$

In der beispielhaften Abbildung wird ein dreistufiger, binärer Signalseparator modelliert, welcher durch ein Oszillationssignal $[O_0^F]$ und seinem inversen Verlauf $[O_0^T]$ gespeist wird. Dabei wird das inverse Signal wie folgt definiert:

- $[O_0^T](t) = a - [O_0^F](t)$

Die Variable a stellt die konstante Amplitude des zwischen 0 und a schwingenden Signalverlaufs dar. Für diese konkrete Aufgabenstellung wird a mit dem Wert 1 belegt.

Die Kaskadenstufe i des Signalseparators besteht aus zwei verschränkt katalysierten Reaktionen. Die Spezies des inversen Oszillationsverlaufs $[O_{i-1}^T]$ wird in der vorherigen Kaskadenstufe als Katalysator der Reaktion $[O_i^F]+[O_{i-1}^T] \xrightarrow{k} [O_i^T]+[O_{i-1}^T]$ verwendet. Dadurch wird mittels $[O_{i-1}^T]$ die Spezies $[O_i^T]$ der Folgestufe angereichert und dem zu separierenden Oszillationsverlauf $[O_i^F]$ das notwendige Material entzogen. Aufgrund dieses Zusammenhangs wird die steigende Flanke des inversen Oszillationsverlaufs mit jeder Kaskadenstufe zunehmend steiler, wodurch ein immer schnelleres Umschaltverhalten erreicht wird.

Durch die Verwendung eines Brusselators und eines solchen binären Signalseperators kann das zu triggernde Reaktionsnetzwerk zur Simulation des aufgestellten deterministischen, endlichen Automaten in ein chemisches Digitalcomputermodell integriert werden.

5.3. Prinzip des Reaktionsnetzwerks für einen Automaten

Der prinzipielle Aufbau des Reaktionsnetzwerks basiert, auf den bereits ausgearbeiteten Schaltungen aus den Abbildungen 3, 4, 5 und 6. Diese logischen Schaltungen werden einzeln als Blöcke zusammengefasst. Die Eingänge dieser Blöcke werden mit den für die Eingabesignale (z_0, z_1, z_2, z_3 und b) definierten Spezien verknüpft. Nach entsprechender Verarbeitung innerhalb eines jeden Blocks wird für das entsprechende Signal der logische Zustandsübergang in Form einer weiteren Spezies ausgegeben.

Für die bit-weise Verarbeitung des Eingabewortes werden die einzelnen, berechneten Zustandsübergänge mittels eines getakteten NOR-Flip-Flops an ihre Ausgangssignale zurückgegeben. Dabei wird für jeden Block ein separates Flip-Flop verwendet, für die zuvor die Signaleingänge R und S berechnet werden. Auf Grundlage dieser Signale wird anschließend für jeden Block das Ausgangssignal Q berechnet, welches wiederum mit dem jeweiligen Eingangsignal (z_0, z_1, z_2, z_3, b) verknüpft ist.

Die Berechnung innerhalb der Blöcke findet immer dann statt, wenn der Takt nicht gesetzt ist. Während der Takt gesetzt ist, werden hingegen die Ausgaben der Flip-Flops berechnet und an das entsprechende Eingangssignal zurückgegeben. Dadurch können für jeden Verarbeitungsschritt des Eingabewortes klare Zustandsübergänge definiert werden.

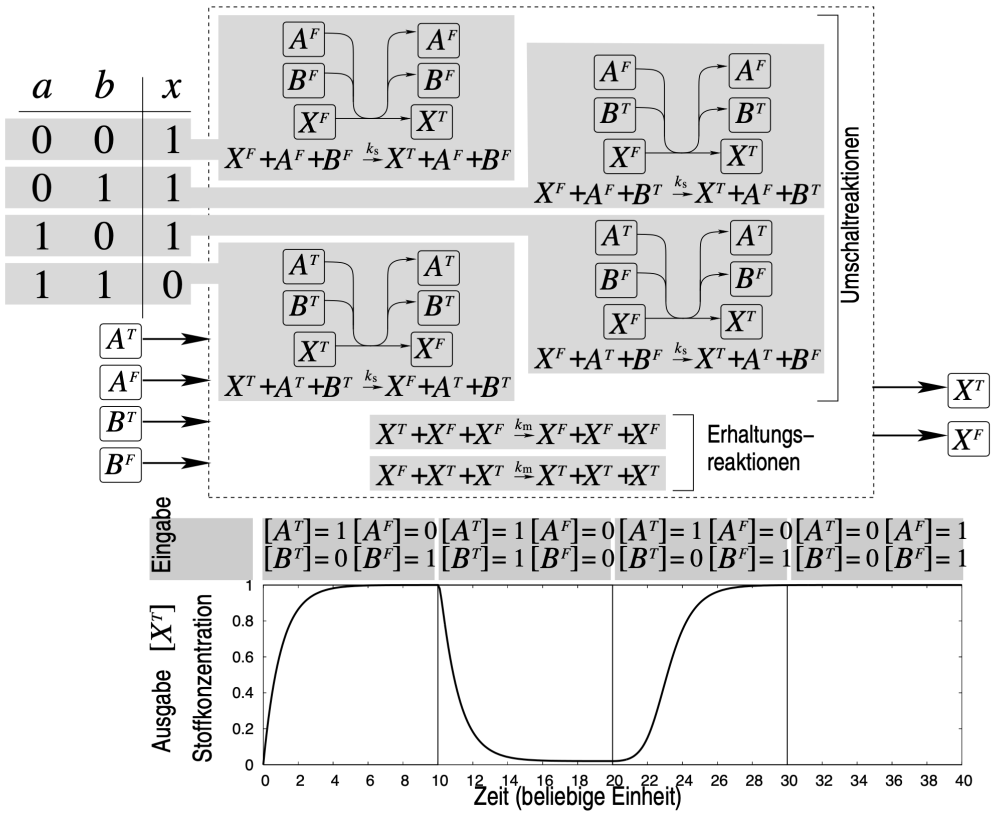
5.4. Aufstellen eines Reaktionsnetzwerks aus logischen Schaltungen

Um aus den einzelnen logischen Schaltungen separate Blöcke für das Reaktionsnetzwerk zu integrieren, werden die in den Schaltungen verwendeten Gatter durch eine Vielzahl ähnlicher Reaktionen abgebildet. Dazu wird für jedes Gatter eine Wahrheitstabelle gebildet, welche die Ergebnisse in Form von Ausgangssignalen für alle möglichen Kombinationen der Eingangssignale wiederspiegeln. Dazu wird für jeden Eintrag einer Wahrheitstabelle wiederum eine explizite Reaktion definiert.

Eine solche Reaktion setzt sich aus den entsprechenden Spezien der Eingangssignale sowie aus der entsprechenden Spezies des Ausgangssignals zusammen. Dabei werden für jedes Signal zwei Spezien definiert, welche den TRUE- und FALSE-Wert des jeweiligen Signals widerspiegeln. Somit wird für jedes Eingangssignal innerhalb einer Reaktion immer die entsprechende FALSE-Spezies verwendet, sofern das entsprechende Signal negiert wurde. Wurde dieses Signal nicht negiert, wird wiederum die entsprechende TRUE-Spezies verwendet. Da die Eingangssignale nur als Katalysatoren der Reaktion verwendet werden, sind die entsprechenden Eingangs-Spezien auf beiden Seiten der Reaktion als Produkte und Edukte in identischer Art und Weise vorhanden.

Neben den Eingangs-Spezien müssen ebenfalls die entsprechenden Ausgangsspezien in der Reaktion integriert werden. Für den Wahrheitswert, welcher aus der entsprechenden Kombination der Eingangssignale entsteht, muss als Produkt der Reaktion die entsprechende Spezies definiert werden. Als Edukt der Reaktion muss wiederum die dazu inverse Spezies des Ausgangssignals definiert werden. Liefert die Eingangskombination beispielsweise den Wahrheitswert TRUE, so muss die TRUE-Spezies des Ausgangssignals als Reaktionsprodukt definiert sein. Die FALSE-Spezies des Ausgangssignals bildet in diesem Fall wiederum ein Edukt der Reaktion.

Abbildung 9 zeigt schematisch die Definition von Reaktionen für ein NAND-Gatter mit zwei Eingangssignalen.



Quelle: „Computer der Natur“

Abbildung 9: künstliche Chemie zur Nachbildung eines NAND-Gatters

Für die Repräsentation aller Gatter der logischen Schaltungen im chemischen Digitalcomputermodell werden pro Gatter je zwei Spezien zur Ausgabe des Ergebnisses benötigt. Weiterhin müssen in Abhängigkeit der Anzahl der Eingabesignale entsprechend viele Reaktionen definiert werden. So müssen für n Eingangssignale 2^n Reaktionen definiert werden.

Um dieses exponentielle Wachstum der Anzahl an Reaktionen zu minimieren, sollten alle Gatter der logischen Schaltungen zerlegt werden. Dabei werden alle Gatter mit mehr als 2 Eingängen durch eine Kaskadierung rekursiv aufgeteilt. Die einzelnen Gatter der Kaskade beinhalten lediglich 2 Eingänge und führen die gleiche logische Operation wie das eigentliche Gatter aus. Dabei werden die Eingangssignale des eigentlichen Gatters innerhalb der ersten Kaskade jeweils paarweise miteinander verknüpft. In jeder weiteren Kaskade werden rekursiv alle weiteren entstehenden Ausgangssignale paarweise miteinander verknüpft, bis nur noch ein Ausgangssignal

entsteht.

Durch Verwendung mehrerer Gatter steigt die Anzahl benötigter Spezien linear. Jedoch wird durch die Verwendung von Gattern mit nur 2 Eingängen das exponentielle Wachstum der Anzahl benötigter Reaktionen minimiert, wodurch das gesamte Reaktionsnetzwerk kompakter abgebildert werden kann.

6. Simulation des Reaktionsnetzwerkes

Das formal definierte Reaktionsnetzwerk soll in *COPASI* (Complex Pathway Simulator) implementiert werden. Dieses Programm ist ein Open-Source-Projekt, welches zur Simulation und Analyse biochemischer Reaktionsnetzwerke dient.

Für das in COPASI implementierte Reaktionsnetzwerk sollen mehrere Fallstudien simuliert werden. Dazu werden verschiedene Eingabeworte unterschiedlicher Länge verwendet, welche für das Eingabesignal b definiert werden. Dabei werden die Ziffern einer jeden Dezimalzahl separat als 4-Bit-Muster kodiert. Die daraus entstehende Bitfolge soll taktweise auf dem Eingabesignal abgebildet werden. Demzufolge soll das entsprechende Signal mit steigender Taktflanke immer dann verändert werden, wenn der Wert von zwei aufeinanderfolgenden Bits unterschiedlich ist. Ein mögliches Eingabesignal für die Simulation der Ziffer 9 kann der Abbildung 10 entnommen werden.

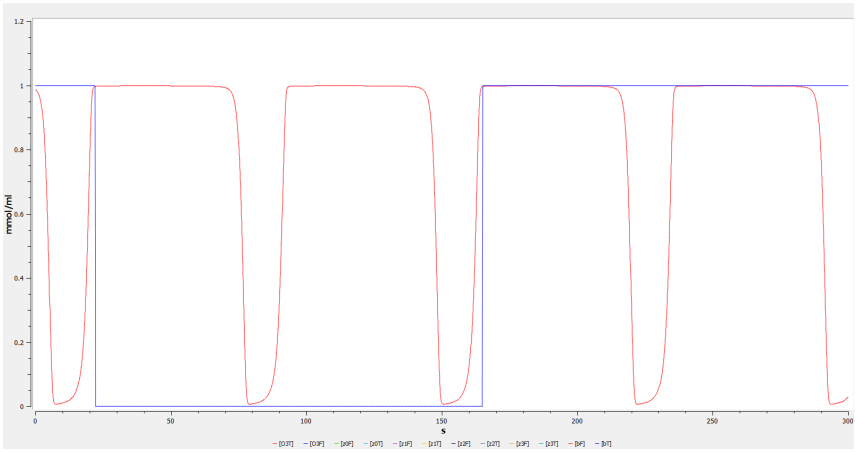


Abbildung 10: Simulation der Dezimalzahl 9 als Eingabewort

Im folgenden werden Fallstudien für das chemische Digitalcomputermodell mit den Eingaben der Dezimalzahlen 0, 7, 9, 30, 37, 867 und 556 aufgestellt.

6.1. Simulation der Dezimalzahl 0 als Eingabewort

Die Dezimalzahl 0 wird im chemischen Digitalcomputermodell als Bitfolge 0000 eingelesen. Anhand der grafischen Darstellung des deterministischen, endlichen Automaten (s. Abbildung 2) lassen sich die folgenden Zustandsübergänge ableiten:

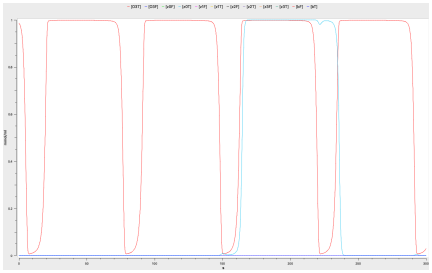
- $q_0 \xrightarrow{0} q_3 \xrightarrow{0} q_6 \xrightarrow{0} q_9 \xrightarrow{0} q_0$

Da die Eingabe im Finalzustand q_0 endet, wird die eingegebene Dezimalzahl von dem Automaten als Eingabewort akzeptiert. Dieses Verhalten entspricht den Erwartungen, da die Dezimalzahl 0 zur akzeptierten Sprache des Automaten gehört.

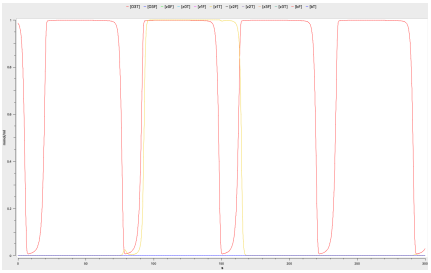
Die Zustandsabfolge wird in Tabelle (3) abgebildet. Das Netzwerkverhalten der einzelnen Bits im chemischen Digitalcomputermodell wird wiederum in Abbildung 11 dargestellt.

Eingabe	0	0	0	0
z_0	0	0	1	0
z_1	0	1	0	0
z_2	1	1	0	0
z_3	1	0	1	0
Q	q_3	q_6	q_9	q_0

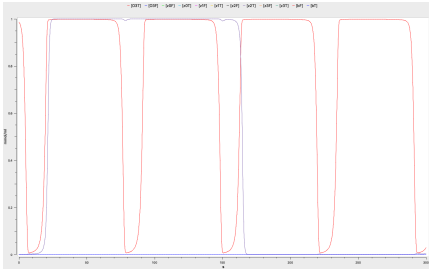
Tabelle 3: Abfolge der Eingabe



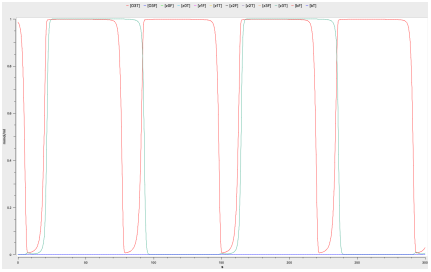
(a) Netzwerkverhalten von z_0



(b) Netzwerkverhalten von z_1



(c) Netzwerkverhalten von z_2



(d) Netzwerkverhalten von z_3

Abbildung 11: Netzwerkverhalten bei Eingabe $b = 0000_2$ (dezimal: 0)

6.2. Simulation der Dezimalzahl 7 als Eingabewort

Die Dezimalzahl 7 wird im chemischen Digitalcomputermodell durch die Bitfolge 0111 repräsentiert. Das chemische Digitalcomputermodell durchläuft auf Grundlage dieser Eingabe die folgende Zustandsabfolge:

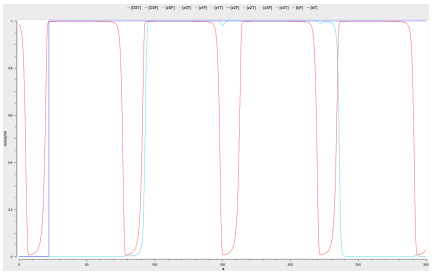
- $q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_8 \xrightarrow{1} q_9 \xrightarrow{1} q_2$

Da die Eingabe nicht in dem Finalzustand q_0 endet, wird das Wort nicht akzeptiert. Das entspricht dem erwarteten Ergebnis, da 7 eine nicht durch 3 teilbare Dezimalzahl ist.

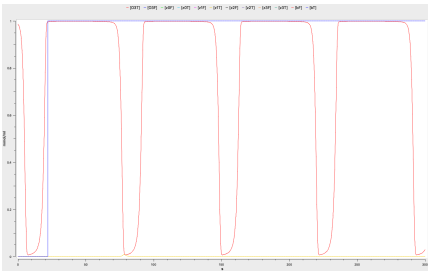
Die Zustandsabfolge wird in Tabelle (4) abgebildet. Das Netzwerkverhalten der einzelnen Bits im chemischen Digitalcomputermodell wird wiederum in Abbildung 12 dargestellt.

Eingabe	0	1	1	1
z_0	0	1	1	0
z_1	0	0	0	0
z_2	1	0	0	1
z_3	1	0	1	0
Q	q_3	q_8	q_9	q_2

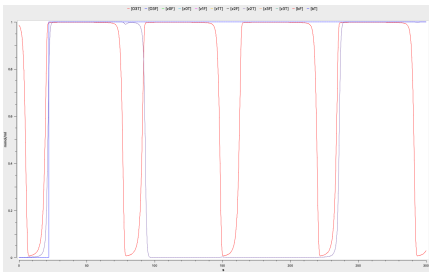
Tabelle 4: Abfolge der Eingabe



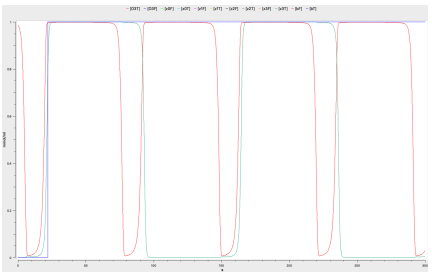
(a) Netzwerkverhalten von z_0



(b) Netzwerkverhalten von z_1



(c) Netzwerkverhalten von z_2



(d) Netzwerkverhalten von z_3

Abbildung 12: Netzwerkverhalten bei Eingabe $b = 0111_2$ (dezimal: 7)

6.3. Simulation der Dezimalzahl 9 als Eingabewort

Die Dezimalzahl 9 wird als Bitfolge 1001 dargestellt. Da diese Dezimalzahl durch 3 teilbar ist, endet der Automat bei Eingabe dieses Wortes im Finalzustand q_0. Dementsprechend gehört die Dezimalzahl 9 zur Menge der akzeptierten Wörter des Automaten.

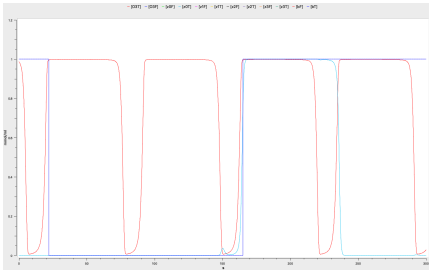
Die konkrete Zustandsabfolge entspricht dem folgenden Ausdruck:

- $q_0 \xrightarrow{1} q_4 \xrightarrow{0} q_7 \xrightarrow{0} q_{10} \xrightarrow{1} q_0$

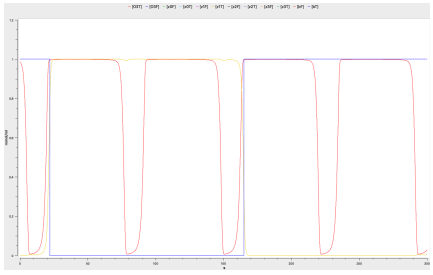
Die Zustandsabfolge wird in Tabelle (5) abgebildet. Das Netzwerkverhalten der einzelnen Bits im chemischen Digitalcomputermodell wird wiederum in Abbildung 13 dargestellt.

Eingabe	1	0	0	1
z_0	0	0	1	0
z_1	1	1	0	0
z_2	0	1	1	0
z_3	0	1	0	1
Q	q_4	q_7	q_{10}	q_1

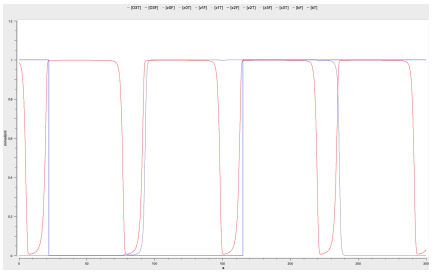
Tabelle 5: Abfolge der Eingabe



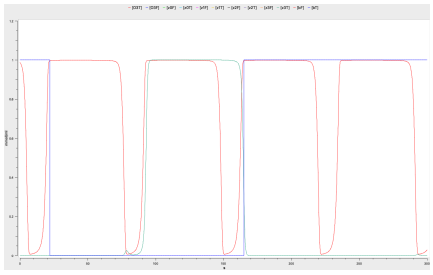
(a) Netzwerkverhalten von z_0



(b) Netzwerkverhalten von z_1



(c) Netzwerkverhalten von z_2



(d) Netzwerkverhalten von z_3

Abbildung 13: Netzwerkverhalten bei Eingabe $b = 1001_2$ (dezimal: 9)

6.4. Simulation der Dezimalzahl 30 als Eingabewort

Bei dem Eingabewort 30 handelt es sich um eine zweistellige Dezimalzahl, welche folglich durch 8 Bits kodiert wird, da jede Dezimalstelle durch ein 4-Bit-Muster kodiert wird. Dem entsprechend wird diese Dezimalzahl durch die Bitfolge 00110000 repräsentiert.

Aufgrund der Länge des Eingabewortes werden im Vergleich zu den vorherigen

Fällen entsprechend mehr Zustände durchlaufen. Die konkrete Zustandsabfolge ergibt sich wie folgt:

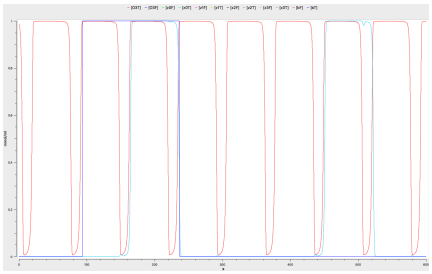
- $q_0 \xrightarrow{0} q_3 \xrightarrow{0} q_6 \xrightarrow{1} q_{10} \xrightarrow{1} q_0 \xrightarrow{0} q_3 \xrightarrow{0} q_6 \xrightarrow{0} q_9 \xrightarrow{0} q_0$

Da der Automat nach Eingabe des Wortes im Zustand q_0 endet, zählt die Dezimalzahl 30 zur Menge der akzeptierten Wörter des Automaten. Dieses Verhalten entspricht den Erwartungen.

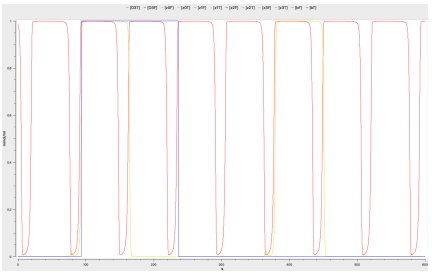
Die Zustandsabfolge wird in Tabelle 6 abgebildet. Das Netzwerkverhalten der einzelnen Bits im chemischen Digitalcomputermodell wird wiederum in Abbildung 14 dargestellt.

Eingabe	0	0	1	1	0	0	0	0
z_0	0	0	1	0	0	0	1	0
z_1	0	1	0	0	0	1	0	0
z_2	1	1	1	0	1	1	0	0
z_3	1	0	0	0	1	0	1	0
Q	q_3	q_6	q_{10}	q_0	q_3	q_6	q_9	q_0

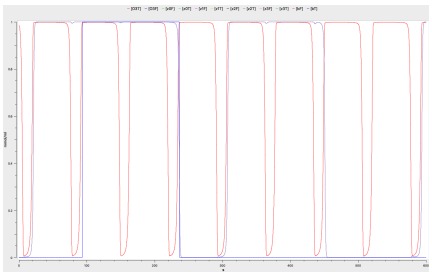
Tabelle 6: Abfolge der Eingabe



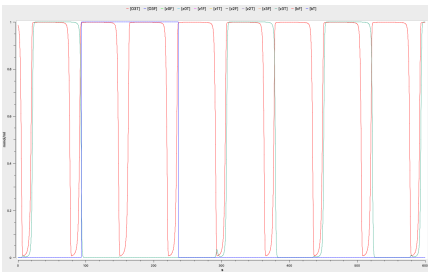
(a) Netzwerkverhalten von z_0



(b) Netzwerkverhalten von z_1



(c) Netzwerkverhalten von z_2



(d) Netzwerkverhalten von z_3

Abbildung 14: Netzwerkverhalten bei Eingabe $b = 0011\,0000_2$ (dezimal: 30)

6.5. Simulation der Dezimalzahl 37 als Eingabewort

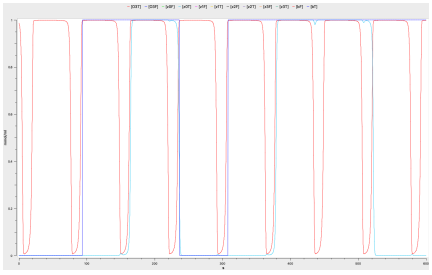
Analog zur 30 handelt es sich bei der 37 ($0011\,0111_2$) ebenfalls um eine zweistellige Dezimalzahl. Dementsprechend wird diese Zahl als die 8-stellige Bitfolge 00110111 in den Automaten des chemischen Digitalcomputermodells eingelesen. Da die 37 nicht durch 3 teilbar ist, sollte der Automat dieses Wort nicht akzeptieren. Die konkrete Zustandsabfolge sieht demnach wie folgt aus:

- $q_0 \xrightarrow{0} q_3 \xrightarrow{0} q_6 \xrightarrow{1} q_{10} \xrightarrow{1} q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_8 \xrightarrow{1} q_9 \xrightarrow{1} q_2$

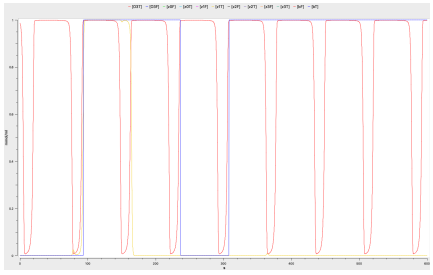
Die Zustandsabfolge wird in Tabelle 7 abgebildet. Das Netzwerkverhalten der einzelnen Bits im chemischen Digitalcomputermodell wird wiederum in Abbildung 15 dargestellt.

Eingabe	0	0	1	1	0	1	1	1
z_0	0	0	1	0	0	1	1	0
z_1	0	1	0	0	0	0	0	0
z_2	1	1	1	0	1	0	0	1
z_3	1	0	0	0	1	0	1	0
Q	q_3	q_6	q_{10}	q_0	q_3	q_8	q_9	q_2

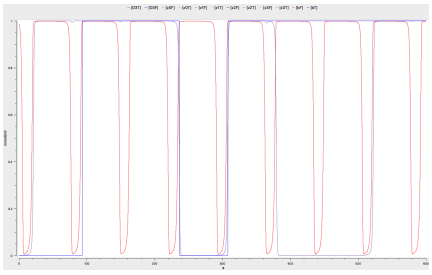
Tabelle 7: Abfolge der Eingabe



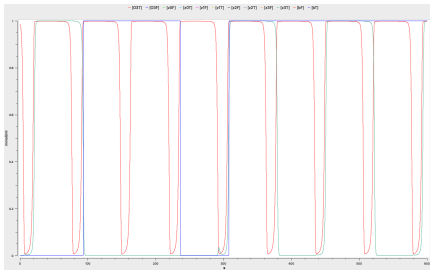
(a) Netzwerkverhalten von z_0



(b) Netzwerkverhalten von z_1



(c) Netzwerkverhalten von z_2



(d) Netzwerkverhalten von z_3

Abbildung 15: Netzwerkverhalten bei Eingabe $b = 0011\,0111_2$ (dezimal: 37)

6.6. Simulation der Dezimalzahl 867 als Eingabewort

Bei der 867 handelt es sich um eine dreistellige Dezimalzahl. Dadurch entsteht für die Eingabe dieses Wortes für den Automaten des chemischen Digitalcomputermodells die Bitfolge 100001100111 mit einer Länge von 12 Zeichen.

Die konkrete Zustandsabfolge entspricht dem folgenden Verhalten:

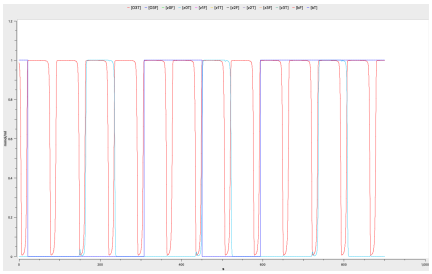
- $q_0 \xrightarrow{1} q_4 \xrightarrow{0} q_7 \xrightarrow{0} q_{10} \xrightarrow{0} q_1 \xrightarrow{0} q_4 \xrightarrow{1} q_6 \xrightarrow{1} q_{10} \xrightarrow{0} q_1 \xrightarrow{0} q_4 \xrightarrow{1} q_6 \xrightarrow{1} q_{10} \xrightarrow{1} q_0$

Das Wort endet im Zustand q_0, welcher ein Finalzustand ist. Dementsprechend ist die Zahl 867 durch 3 teilbar Zahl.

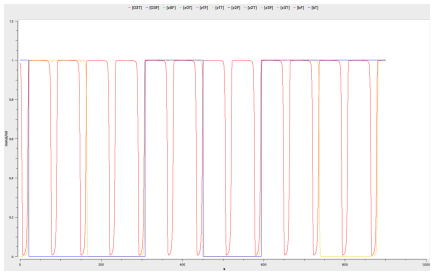
Die Zustandsabfolge wird in Tabelle 8 abgebildet. Das Netzwerkverhalten der einzelnen Bits im chemischen Digitalcomputermodell wird wiederum in Abbildung 16 dargestellt.

Eingabe	1	0	0	0	0	1	1	0	0	1	1	1
z_0	0	0	1	0	0	0	1	0	0	0	1	0
z_1	1	1	0	0	0	1	0	0	1	1	0	0
z_2	0	1	1	0	1	1	1	0	0	1	1	0
z_3	0	1	0	1	0	0	0	1	0	0	0	0
Q	q_4	q_7	q_{10}	q_1	q_4	q_6	q_{10}	q_1	q_4	q_6	q_{10}	q_0

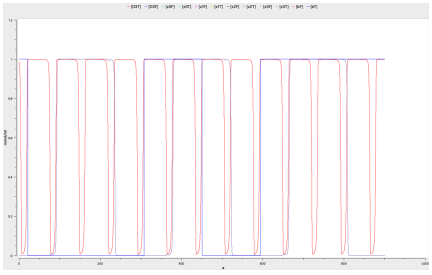
Tabelle 8: Abfolge der Eingabe



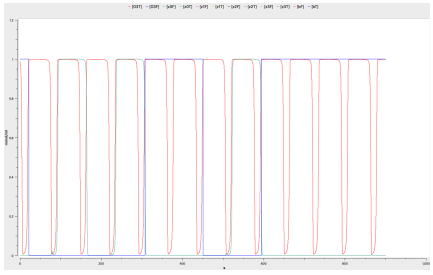
(a) Netzwerkverhalten von z_0



(b) Netzwerkverhalten von z_1



(c) Netzwerkverhalten von z_2



(d) Netzwerkverhalten von z_3

Abbildung 16: Netzwerkverhalten bei Eingabe $b = 1000\,0110\,0111_2$ (dezimal: 867)

6.7. Simulation der Dezimalzahl 556 als Eingabewort

Da die Dezimalzahl 556 ebenfalls dreistellig ist, wird sie für den Automaten als zwölfstellige Bitfolge kodiert. Die konkrete eingebene Bitfolge lautet 010101010110. Da die Dezimalzahl 556 nicht durch 3 teilbar ist, endet sie nicht im Endzustand q_0.

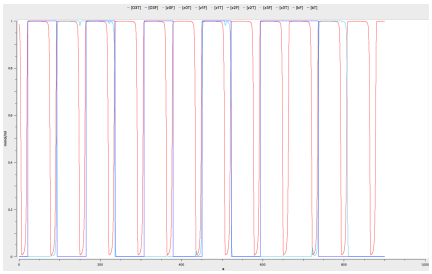
Die konkrete Zustandsabfolge ist wie folgt definiert:

- $q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_8 \xrightarrow{0} q_{11} \xrightarrow{1} q_1 \xrightarrow{0} q_4 \xrightarrow{1} q_6 \xrightarrow{0} q_9 \xrightarrow{1} q_2 \xrightarrow{0} q_5 \xrightarrow{1} q_7 \xrightarrow{1} q_{11} \xrightarrow{0} q_2$

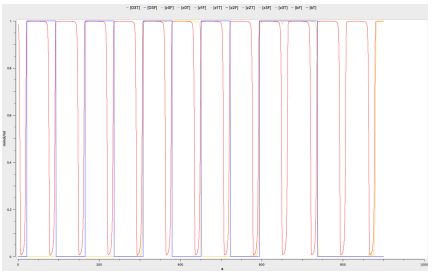
Die Zustandsabfolge wird in Tabelle 9 abgebildet. Das Netzwerkverhalten der einzelnen Bits im chemischen Digitalcomputermodell wird wiederum in Abbildung 17 dargestellt.

Eingabe	0	1	0	1	0	1	0	1	0	1	1	0
z_0	0	1	1	0	0	0	1	0	0	0	1	0
z_1	0	0	0	0	0	1	0	0	1	1	0	0
z_2	1	0	1	0	1	1	0	1	0	1	0	1
z_3	1	0	1	1	0	0	1	0	1	1	1	0
Q	q_3	q_8	q_{11}	q_1	q_4	q_6	q_9	q_2	q_5	q_7	q_{11}	q_2

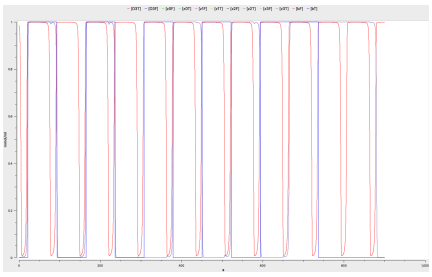
Tabelle 9: Abfolge der Eingabe



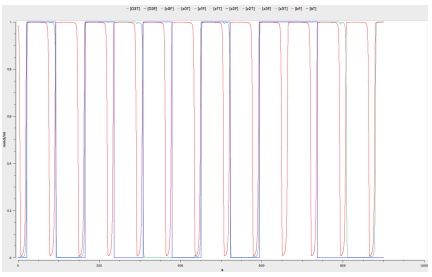
(a) Netzwerkverhalten von z_0



(b) Netzwerkverhalten von z_1



(c) Netzwerkverhalten von z_2



(d) Netzwerkverhalten von z_3

Abbildung 17: Netzwerkverhalten bei Eingabe $b = 0101\ 0101\ 0110_2$ (dezimal: 556)

7. Diskussion und Fazit

Anhand der Fallstudien in Kapitel 6 wird ersichtlich, dass der im chemischen Digitalcomputermodell implementierte Automat vollständig funktionsfähig ist und korrekterweise nur genau die Zahlen akzeptiert, welche durch 3 teilbar sind. Das Verhalten des Automaten simuliert auf Grundlage konkreter Eingaben das gewünschte Verhalten. Dabei verarbeitet der Automat sogar Eingaben beliebiger Länge zuverlässig und korrekt.

Durch weitere Tests des in COPASI implementierten Automaten konnte festgestellt werden, dass dieser Automat jede 4-stellige Binärzahl ebenfalls richtig verarbeitet und dementsprechend auch die entsprechenden Bitmuster der Zahlen 12 (1100_2) und 15 (1111_2) akzeptiert. Prinzipiell liegt diese Eigenschaft in dem Aufbau des aufgestellten deterministischen, endlichen Automaten begründet. Der Automat addiert für jede 4-stellige Binärzahl die Restklassen $\bmod 3$ der entsprechenden Zweierpotenzen, sofern an der jeweiligen Stelle der Binärzahl eine 1 enthalten ist. Dadurch ist es nicht nur möglich, schrittweise einstellige Dezimalzahlen zu überprüfen, sondern auch eben jene Dezimalzahlen, welche ebenfalls durch vierstellige Binärzahlen darstellbar sind.

Ebenfalls ist es möglich, mehrere vierstellige Binärzahlen mittels dieses Modells $\bmod 3$ zu addieren. Mithilfe des Automaten ist es möglich, die Restklassen vierstelliger Binärzahlen zu ermitteln. Würden nacheinander mehrere solcher vierstelligen Binärzahlen den Automaten durchlaufen, würden dementsprechend diese Restklassen miteinander addiert werden.

Würden beispielsweise die Binärzahlen der Zahlen 11 (1011_2) und 13 (1101_2) nacheinander den Automaten durchlaufen, müsste der Automat die Bitfolge 1011 1101 verarbeiten. Dabei würde der Automat im Endzustand q_0 enden und dementsprechend die Eingabe akzeptieren. Um diese These zu stützen, sollte dieses Verhalten jedoch konkreter analysiert und auf der Grundlage der Mathematik gestützt werden.

I. Tabellenverzeichnis

II. Abbildungsverzeichnis

Literatur

[1] Thomas Hinze. „Computer der Natur“. In: (2013). URL: www.molecular-computing.de.

A. Anlagen

Q	z_0	z_1	z_2	z_3
q_0	0	0	0	0
q_1	0	0	0	1
q_2	0	0	1	0
q_3	0	0	1	1
q_4	0	1	0	0
q_5	0	1	0	1
q_6	0	1	1	0
q_7	0	1	1	1
q_8	1	0	0	0
q_9	1	0	0	1
q_{10}	1	0	1	0
q_{11}	1	0	1	1

Tabelle 10: Zustände

a_k	b_1	b_2	b_3	b_4	DEZ(a_k)
a_0	0	0	0	0	0
a_1	0	0	0	1	1
a_2	0	0	1	0	2
a_3	0	0	1	1	3
a_4	0	1	0	0	4
a_5	0	1	0	1	5
a_6	0	1	1	0	6
a_7	0	1	1	1	7
a_8	1	0	0	0	8
a_9	1	0	0	1	9

Tabelle 11: Eingabealphabet

	Q					Q'			
	z_0	z_1	z_2	z_3	b	z'_0	z'_1	z'_2	z'_3
q_0	0	0	0	0	0	0	0	1	1
	0	0	0	0	1	0	1	0	0
q_1	0	0	0	1	0	0	1	0	0
	0	0	0	1	1	0	1	0	1
q_2	0	0	1	0	0	0	1	0	1
	0	0	1	0	1	0	0	1	1
q_3	0	0	1	1	0	0	1	1	0
	0	0	1	1	1	1	0	0	0
q_4	0	1	0	0	0	0	1	1	1
	0	1	0	0	1	0	1	1	0
q_5	0	1	0	1	0	1	0	0	0
	0	1	0	1	1	0	1	1	1
q_6	0	1	1	0	0	1	0	0	1
	0	1	1	0	1	1	0	1	0
q_7	0	1	1	1	0	1	0	1	0
	0	1	1	1	1	1	0	1	1
q_8	1	0	0	0	0	1	0	1	1
	1	0	0	0	1	1	0	0	1
q_9	1	0	0	1	0	0	0	0	0
	1	0	0	1	1	0	0	1	0
q_{10}	1	0	1	0	0	0	0	0	1
	1	0	1	0	1	0	0	0	0
q_{11}	1	0	1	1	0	0	0	1	0
	1	0	1	1	1	0	0	0	1

Tabelle 12: Zustandsübergangstabelle