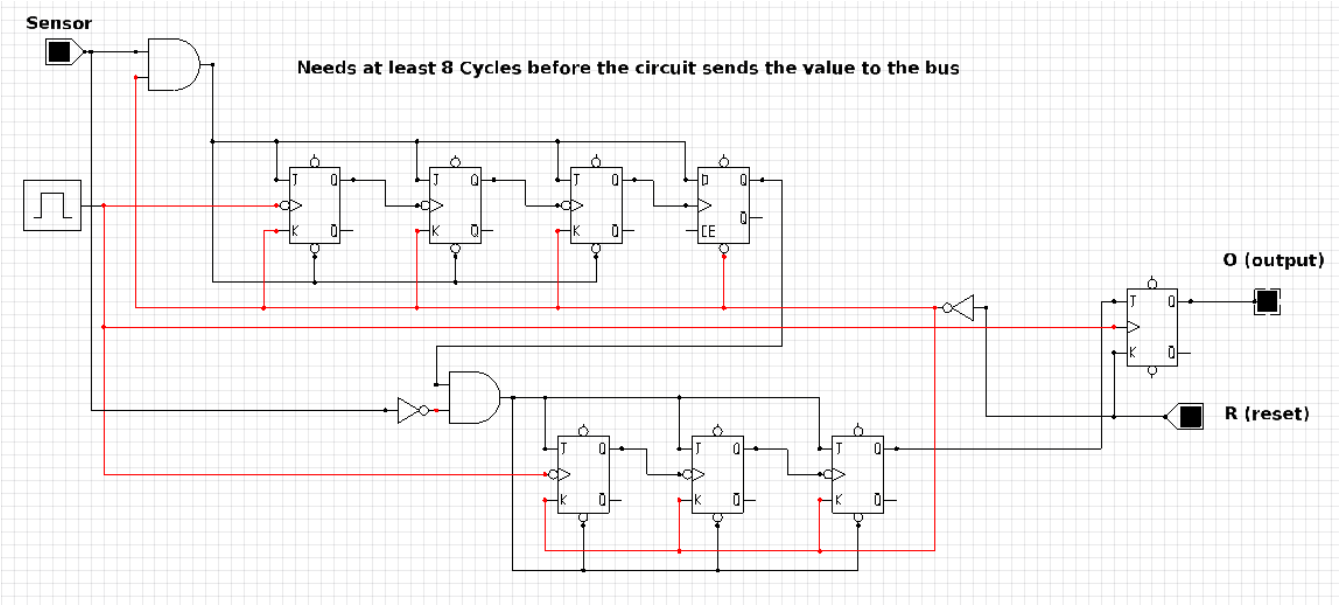
Inhaltsverzeichnis

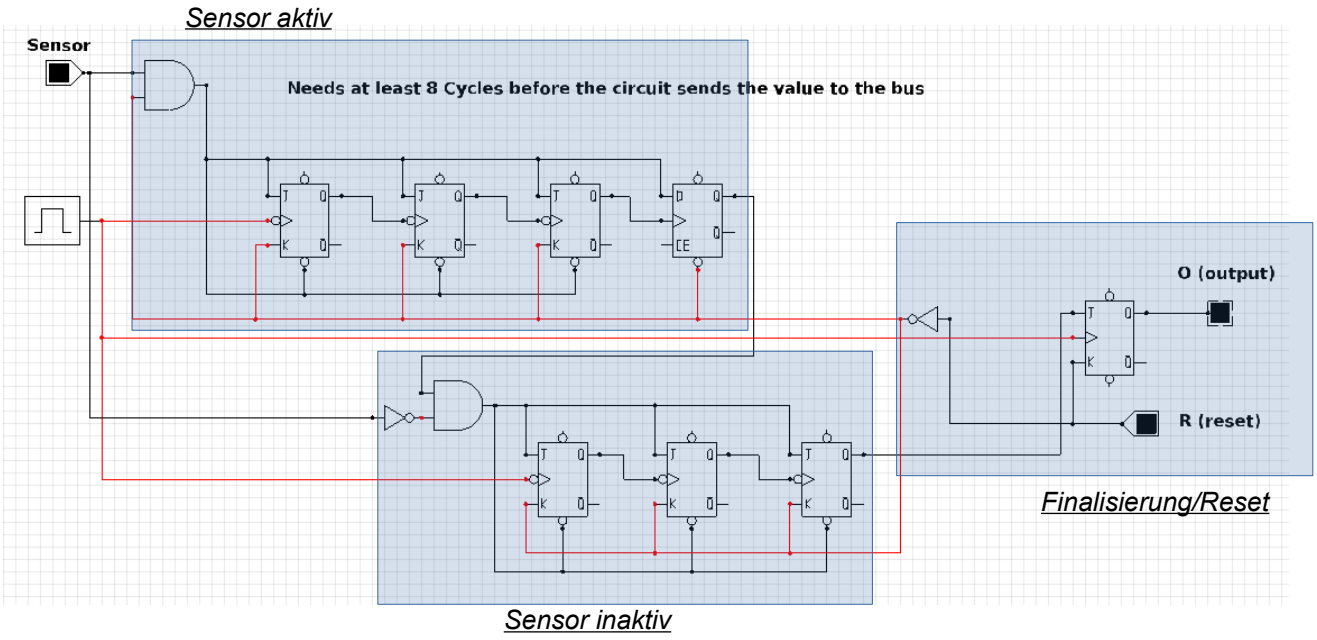
Ich stelle auch alle in diesem Dokument verwendeten Schaltungen und Diagramme zusammen mit der Software zur Verfügung, um sie auf meiner Github-Seite anzuzeigen

Sensorsteuerung

Die Schaltung besteht aus 8 Flip-Flops, wir nutzen die Laufzeitverzögerung aus, um genau 8 Taktzyklen zu warten , natürlich sind die Ressourcen, die zum Ausführen dieser Funktion auf diesem Formular erforderlich sind, kostspielig, aber der Vorteil besteht darin, dass die Schaltung jederzeit zurückgesetzt werden kann.

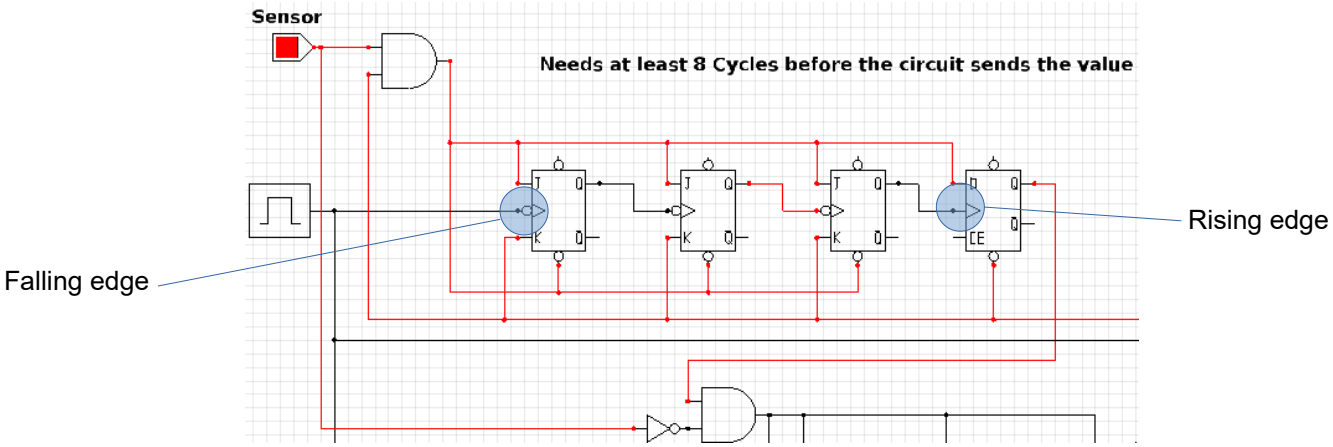


Lassen Sie uns nun ein bisschen mehr ins Detail gehen ,
Die Schaltung selbst besteht aus drei Teilen, *Sensor aktiv*, *Sensor inaktiv* und *Finalisierung/Reset*



Sensor aktiv

Dieser Teil der Schaltung wartet 4 Taktzyklen, bis er den Ausgang der nächsten Stufe passiert, einzige Bedingung ist natürlich, dass der Sencore aktiv oder unterbrochen sein muss

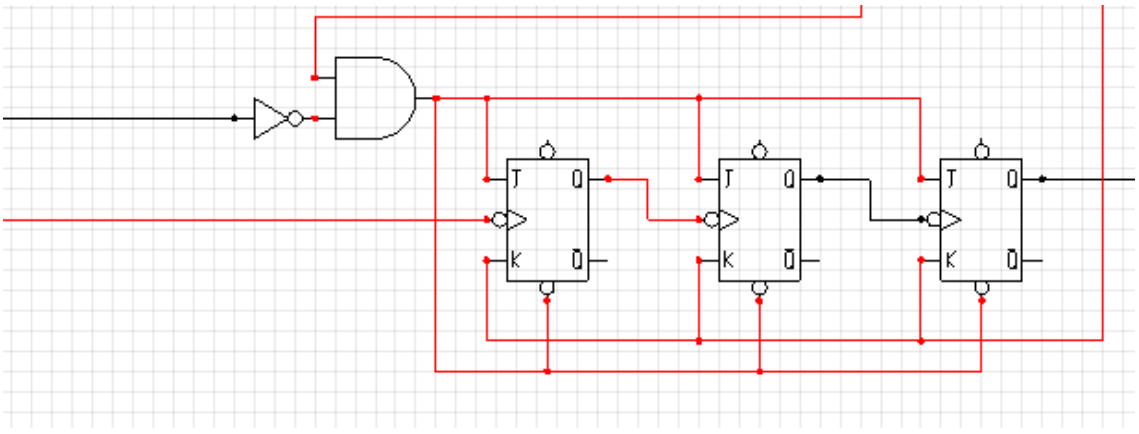


Wir erreichen diesen Warteeffekt, indem wir Schieberegler zum Teilen durch 2 verwenden, ein Flip-Flop-Ausgang ist ein weiterer Flip-Flop-Takt, das heißt, wir müssen doppelt so viele Zyklen für jedes Flip-Flop im Chai warten,

die fallende Flanke (Falling edge) und die steigende Flanke (Rising edge) dienen Synchronisationszwecken, um sicherzustellen, dass sobald ein gültiges Signal am letzten Flip-Flop erkannt wird, es im selben Zyklus weitergegeben wird, Das Signal wird an ein und Gate mit negativem Anschluss an den Sensor weitergegeben

Sensor inaktiv

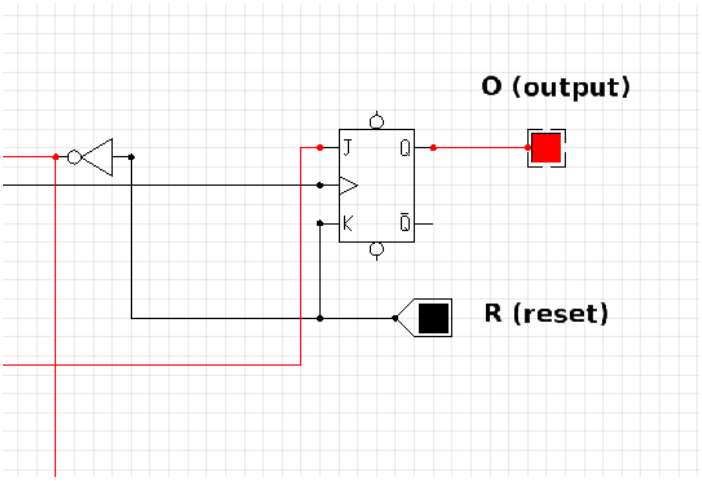
Der nächste Teil der Schaltung wird nur ausgeführt, wenn der Sensor ununterbrochen ist und der vorherige Block beendet ist.



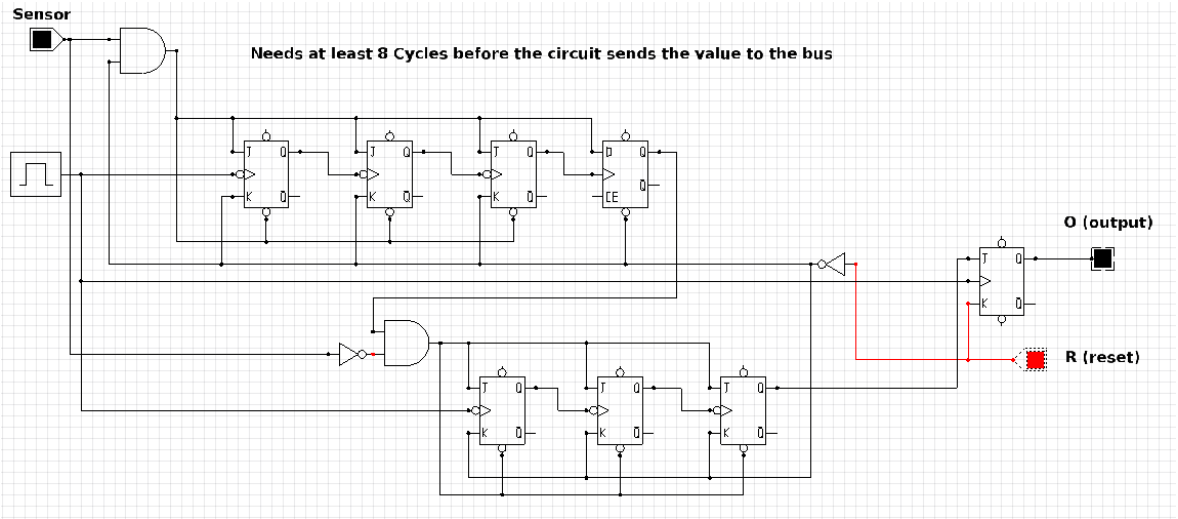
Die Methode ist ähnlich wie im vorherigen Block, der Flip-Flip-Ausgang ist ein weiterer Flip-Flop-Eingang, der einzige Unterschied besteht darin, dass wir den vierten Flip-Flip nicht benötigen, da er verwendet wird, um das Signal an den Bus weiterzuleiten

Finalisierung/Reset

die letzte Stufe nimmt einfach das Signal und gibt es an den Bus weiter, bis das System zurückgesetzt wird



Der mit dem Reset-Schalter verbundene Negator hält alle anderen Flip-Flops aktiv, sobald er eingeschaltet ist, setzt er effektiv das gesamte System zurück, indem er alle Flip-Flops in allen Stufen zurücksetzt

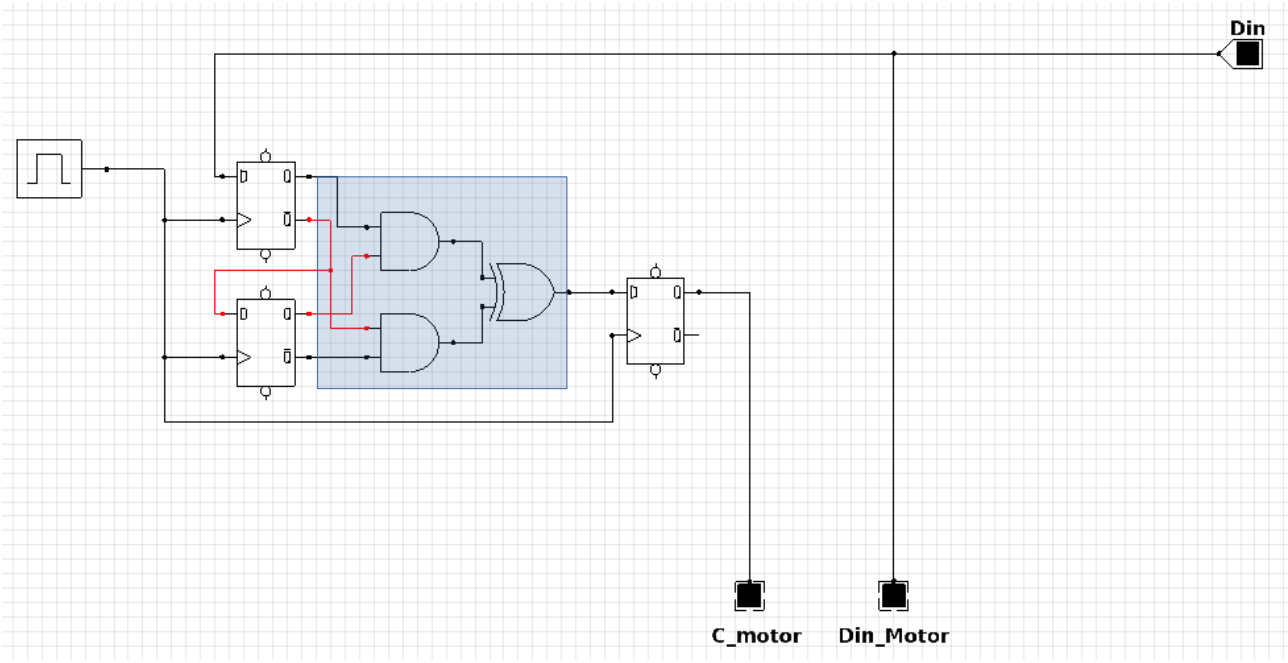


Das System hat auch eine gewisse Fehlertoleranz, denn wenn eine der Arbeitsbedingungen der Stufe nicht erfüllt wird, wird sich die Stufe einfach automatisch zurücksetzen , da das Sensorsignal die Flip-Flops je nach Stufe entweder aktiv oder inaktiv hält

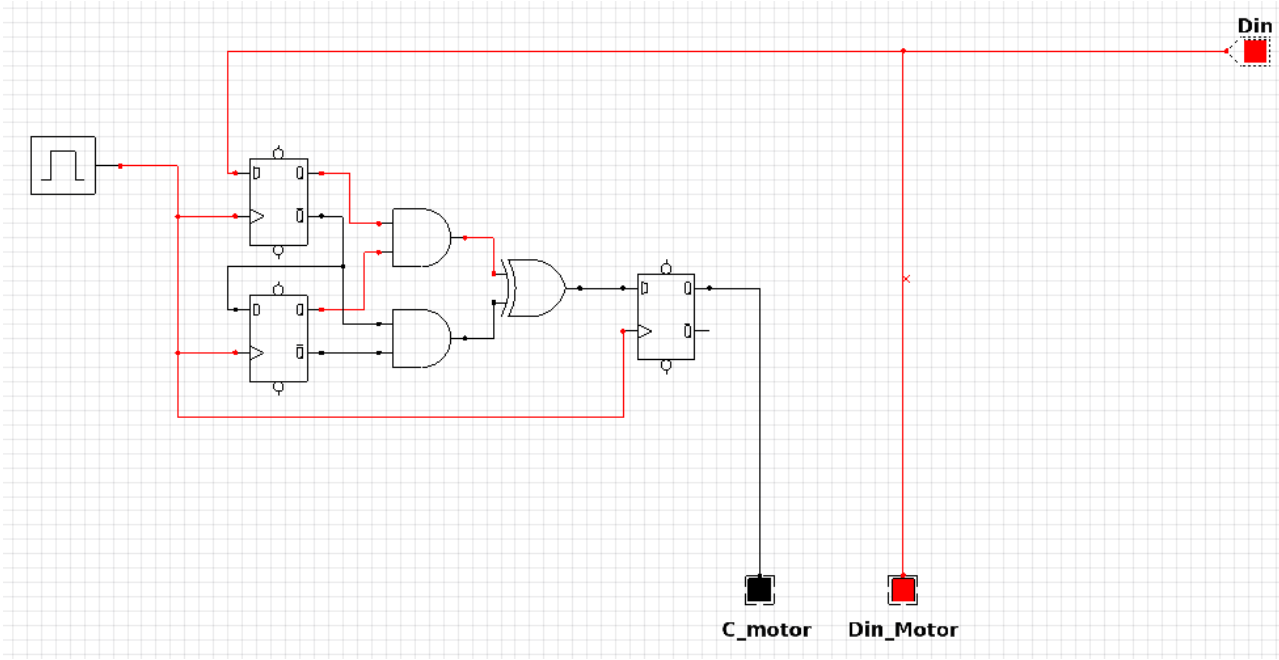
Motorauslösesteuerung

Die Schaltung ist nichts anderes als ein Auslösemechanismus, hier nutzen wir auch die Laufzeitverzögerung der Flip-Flops aus, um sicherzustellen, dass genau ein Taktzyklus einen wahren Wert an die nächste Schaltung weitergibt

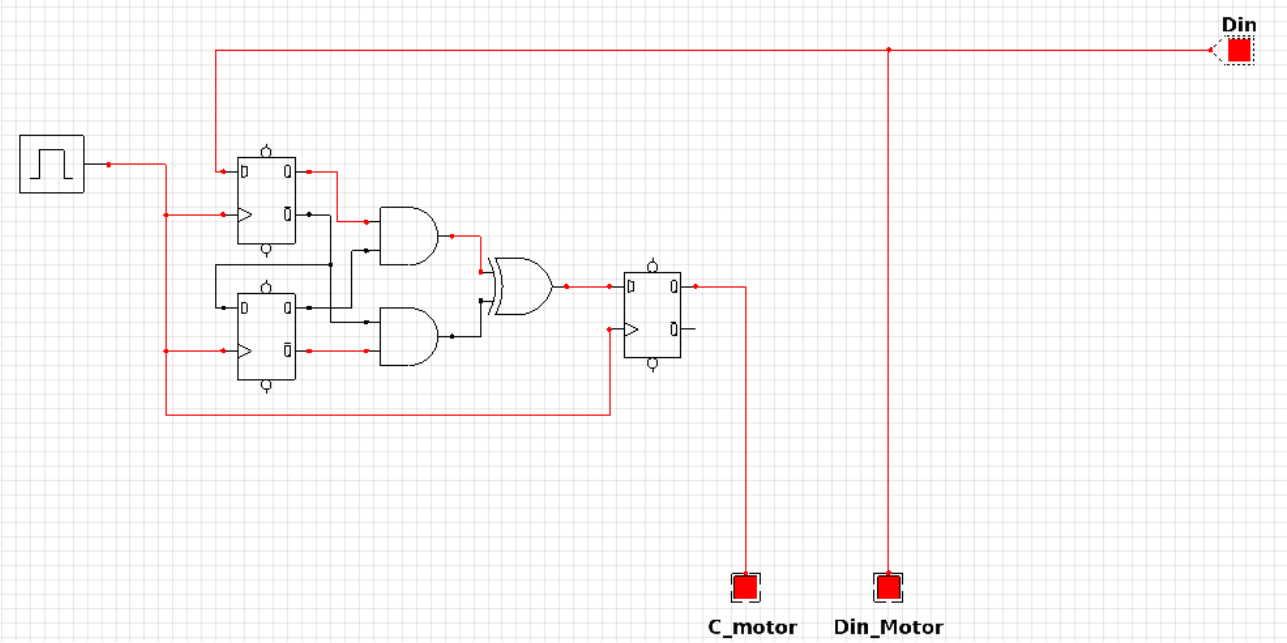
Das System verwendet einfach zwei UND-Gatter für sowohl negative als auch nicht negierte Flip-Flop-Signale und kreuzt beide, wobei es eine UND-Operation an zwei negierten und an zwei negierten Ausgängen durchführt



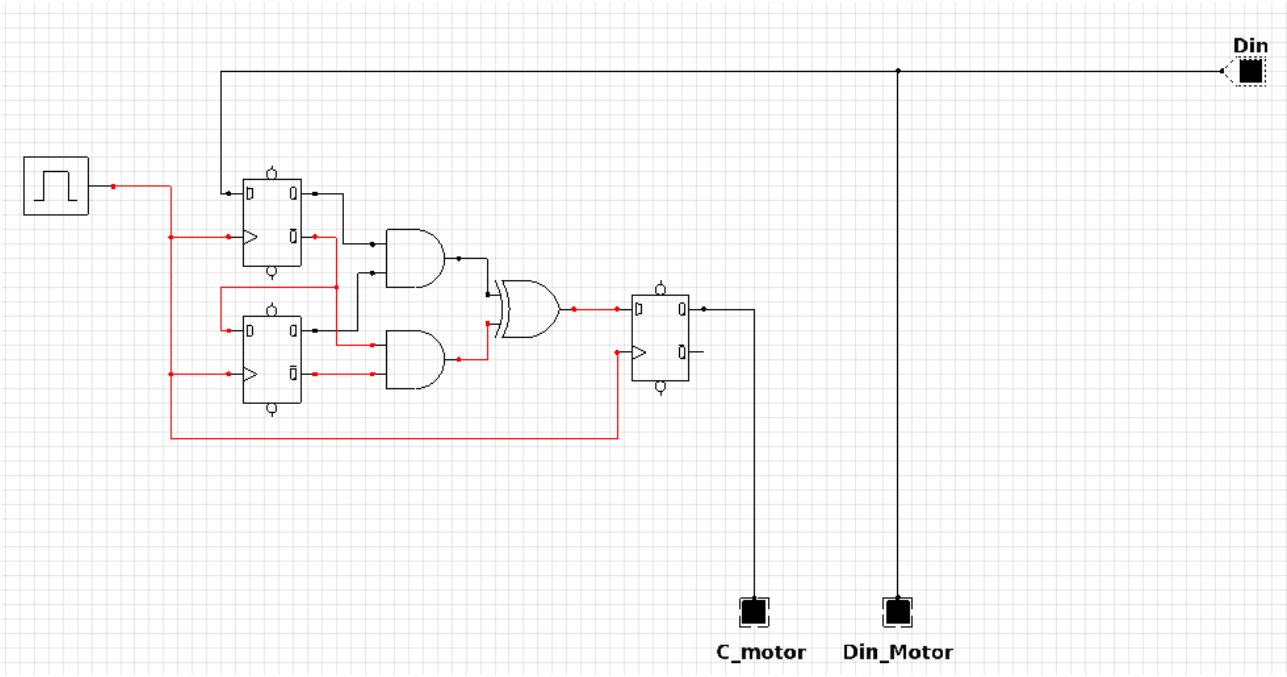
Wenn ein Signal vom Bus empfangen wird, ändert es den Zustand des ersten Flip-Flops, es dauert zusätzlich einen weiteren Taktzyklus, um den Zustand des folgenden Flip-Flops zu ändern



Zwischen diesen Ausbreitungszyklen nimmt das letzte Flip-Flop eine Eingabe vom xor-Gatter und schaltet es für einen Zyklus ein, bis das zweite Flip-Flop aufholt und das Signal blockiert



Das gleiche Verfahren wird verwendet, um ein Signal an die nächste Schaltung weiterzuleiten, wenn der Eingang der vorhandenen Schaltung inaktiv ist, nur umgekehrt, weshalb die negierten Ausgänge der Flip-Flops verwendet werden.



Das System selbst hat eine gewisse Eingangsverzögerung, zwei Zyklen, um genau zu sein, während dieser zwei Zyklen muss sich das Signal auf die nächste Schaltung ausbreiten, während der keine anderen Eingaben verarbeitet werden

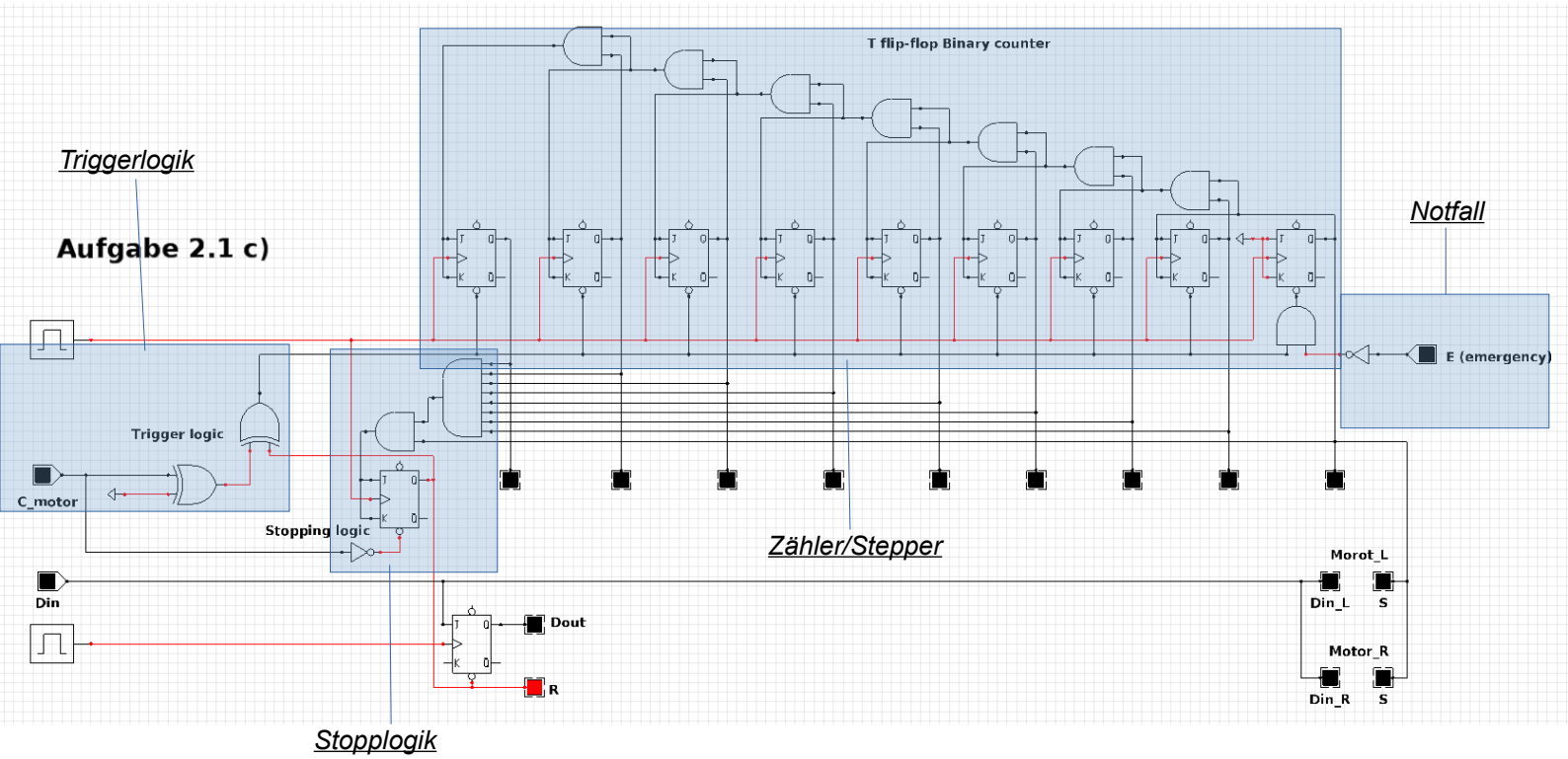
Aufgabe 2.1 c)

Das Ziel dieser Schaltung ist es, die Motoren zu handhaben und einen reibungslosen Betrieb zu gewährleisten, die Schaltung ist nichts anderes als ein 9-Bit-Zähler mit einigen Modifikationen, um die Anzahl der Bits oder Flip-Flops herauszufinden, die wir benötigen, nehmen wir unsere gewünschte Anzahl von auszuführenden Schritten (521) und führen einen Logarithmus von zwei darauf aus

$Log_2(512) = 9$

Eine perfekte Passform, dies ermöglicht es uns, den gesamten Zähler zu durchlaufen, ohne uns darum zu kümmern, was in der Mitte passiert, wenn alle Signale im Zähler wahr sind, die maximale Zahl wird erreicht (511) , dies hilft uns auch, da wir die nicht speichern müssen aktuelle Nummer in ein Register übertragen und später vergleichen, der Zähler kann physikalisch nicht über 512 zählen, daher können wir dies zu unserem Vorteil nutzen, um das System später zu optimieren

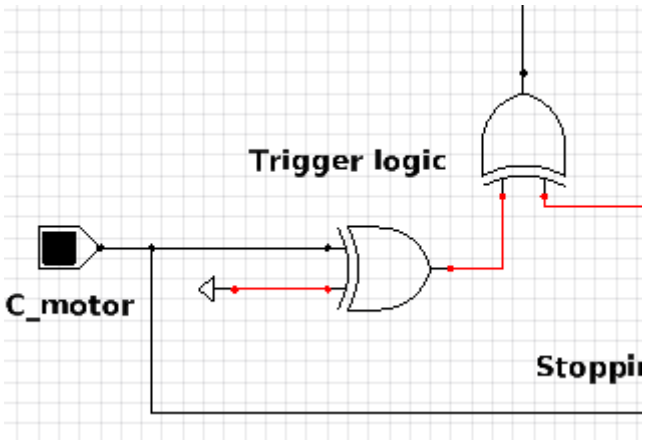
Wir haben das System in 4 kleinere Subsysteme aufgeteilt, *Triggerlogik*, *Zähler/Stepper*, *Stopplogik* und *Notfall*



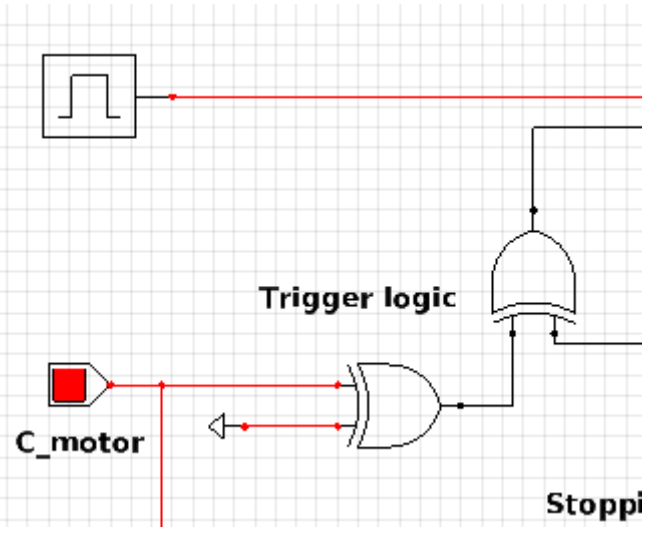
Triggerlogik

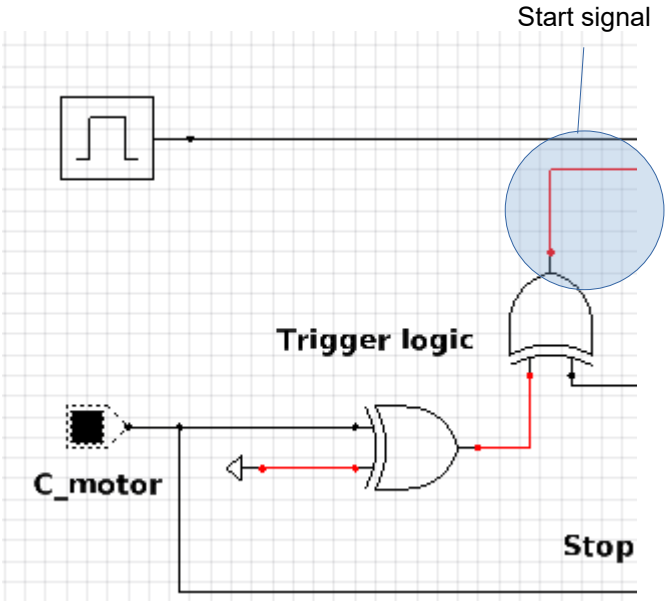
Dieser Teil der Schaltung ist für das Zurücksetzen und Initialisieren des Zählers verantwortlich, er nimmt den Impulseingang von der Schaltung, die wir in der vorherigen Aufgabe behandelt haben, der Eingang ist mit einem XOR-Gatter und einem in der Stopplogik verwendeten Flip-Flop verbunden.

Außerdem teilt es ein weiteres XOR-Gatter mit der Stopplogik, das anfängliche XOR-Gatter ist immer mit einer Stromquelle verbunden, in der vorherigen Aufgabe sendet die für das Senden des Signals verantwortliche Schaltung das Signal in genau einem Zyklusintervall

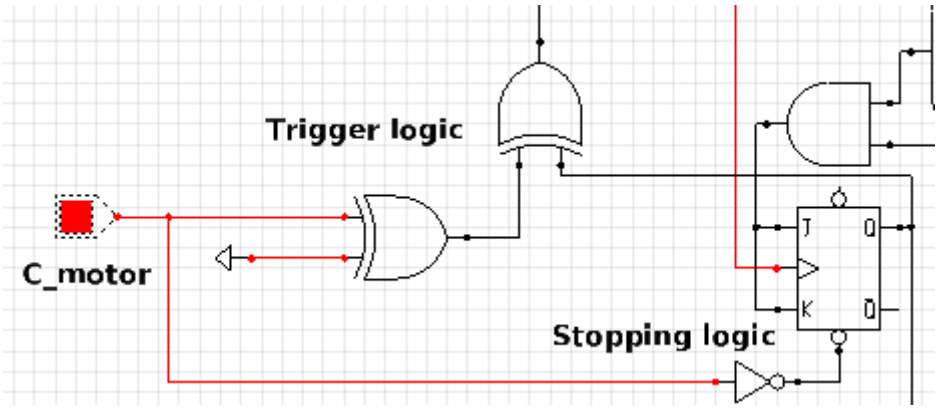


Wir können davon ausgehen, dass das Motorsignal die meiste Zeit inaktiv ist und nur einen Zyklus dauert, gerade genug Zeit, um die Flip-Flops des Zählers zu initialisieren und den Prozess zu starten.





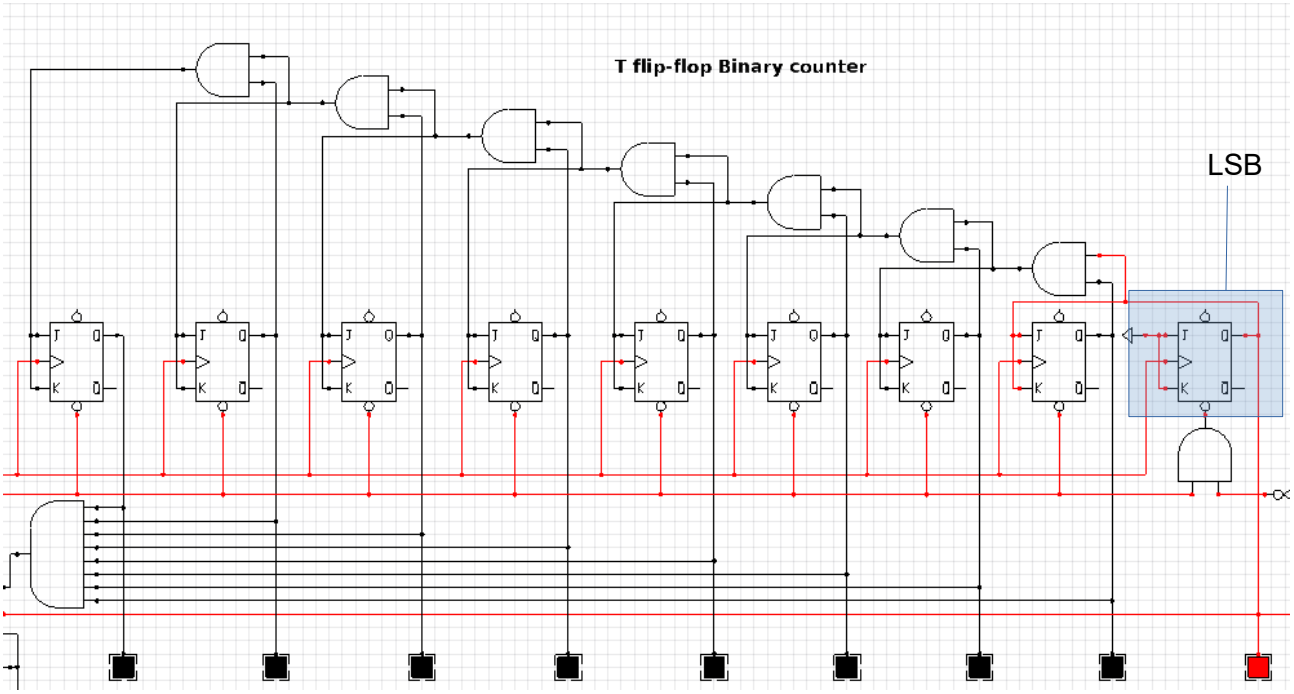
Wenn das Signal aktiv ist, deaktiviert es auch das Stoppsignal-Flip-Flop, sodass nur ein Signal zum Zähler durchgelassen werden kann



Wir werden später näher auf die Stopplogik eingehen, da sie stark mit dem Zähler selbst verbunden ist

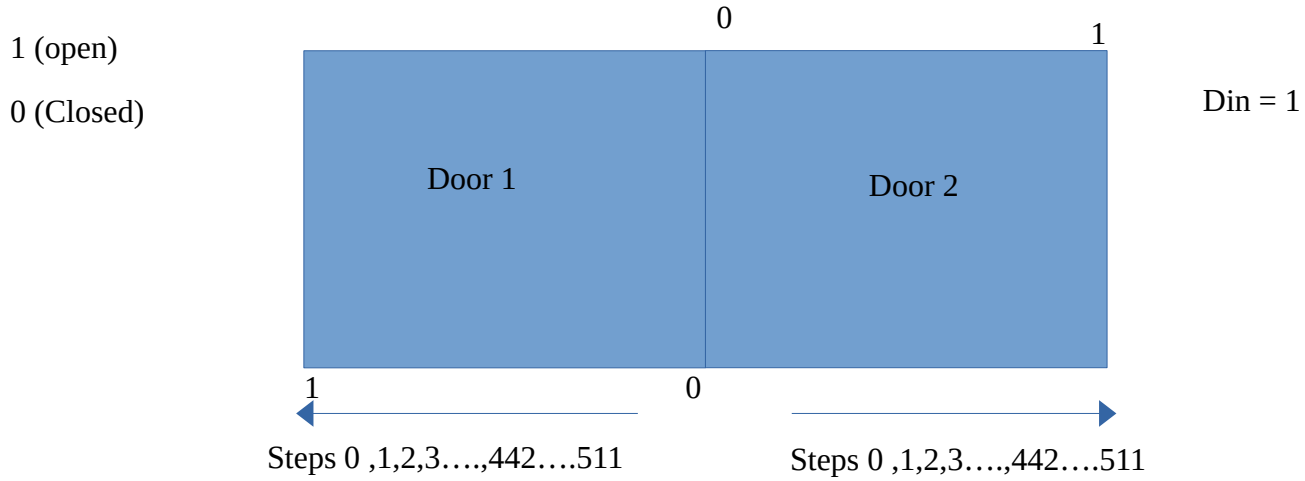
Zähler/Stepper

Der Stepper selbst ist eine Reihe von t-Flip-Flops, die mit und-Gattern verbunden sind, einfach ein Binärzähler, hier verwenden wir die und-Gatter, da wir sie später manipulieren können. Das Prinzip ist, dass der Cointer das nächste Bit einschaltet, wenn alle Bedingungen oder alle Bits, bevor es auch an ist,
dieser Aufbau ist vorteilhaft, da der gesamte Abrechnungsprozess vom niedrigstwertigen Bit abhängig ist



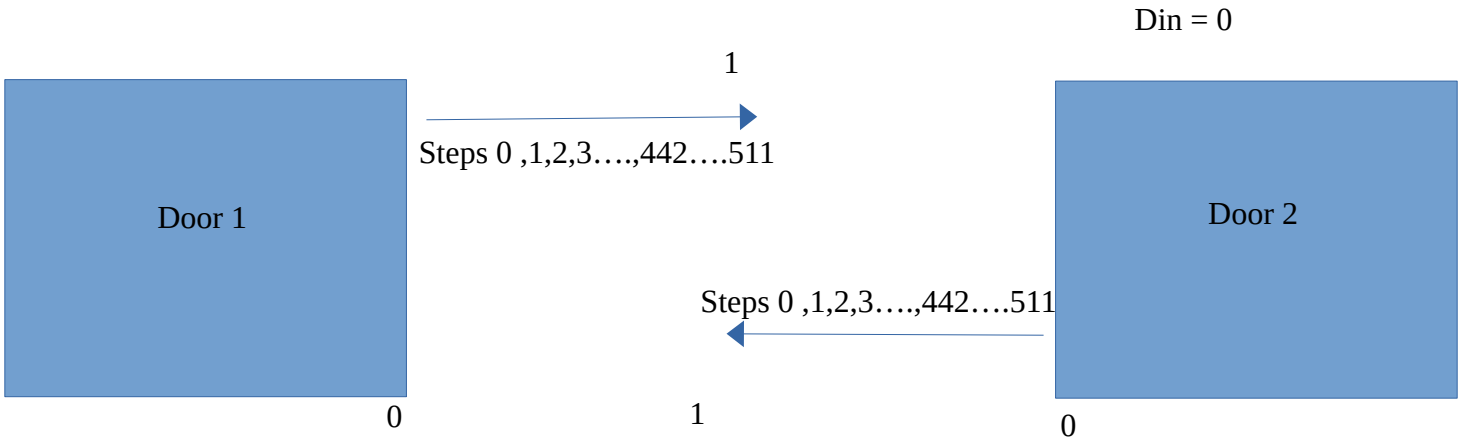
Das LSB ist an eine Stromquelle angeschlossen und sein Eingang ist immer eingeschaltet, da wir unsere Nummer bei jedem Schritt um eins erhöhen, Der Zähler nimmt den Initialisierungseingang von der zuvor besprochenen Triggerschaltung.

Ein weiterer großer Vorteil ist, dass wir keine Berechnungen durchführen müssen, da der Zähler eine feste Größe hat. Wir können seine Funktion zum Öffnen und Schließen der Türen verwenden, indem wir den Vorgang einfach umkehren,
Es ist einfacher gesagt und getan, da die Richtung der Motoren durch den Din-Eingang bestimmt wird.



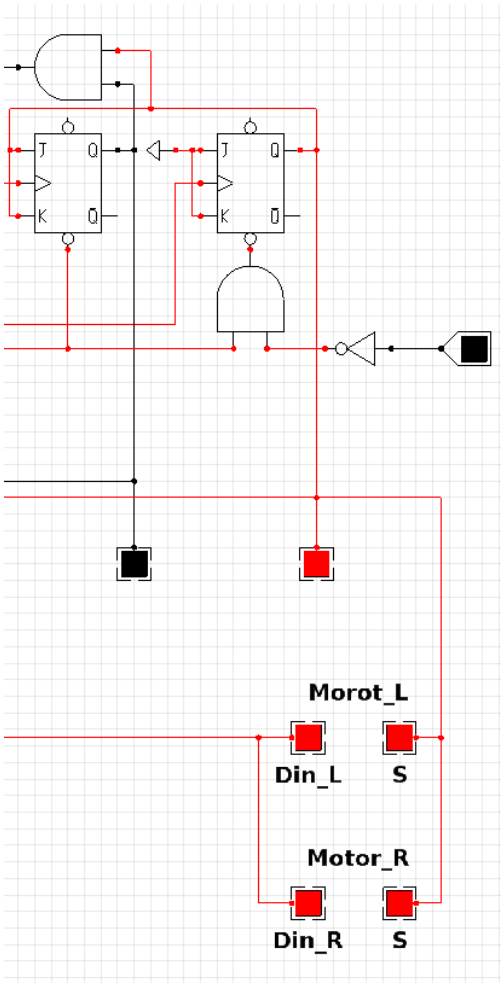
Wenn din auf eins oder null gesetzt ist, setzen wir nur den Zähler neu und geben die Richtungseingabe an die Motoren weiter,

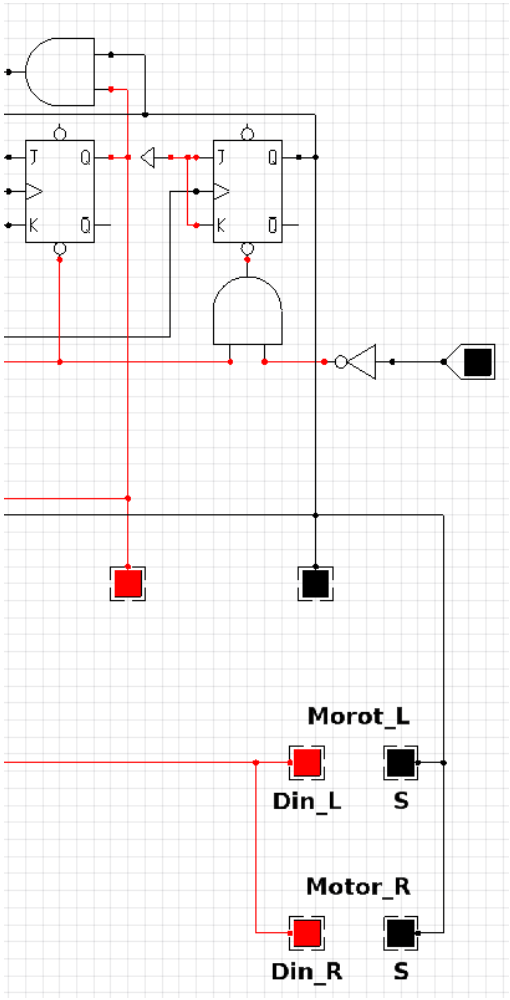
Zu Unsere Schaltung hat dies effektiv die Öffnungs- und Schließrichtung umgekehrt und wieder von 0 bis 511 gezählt



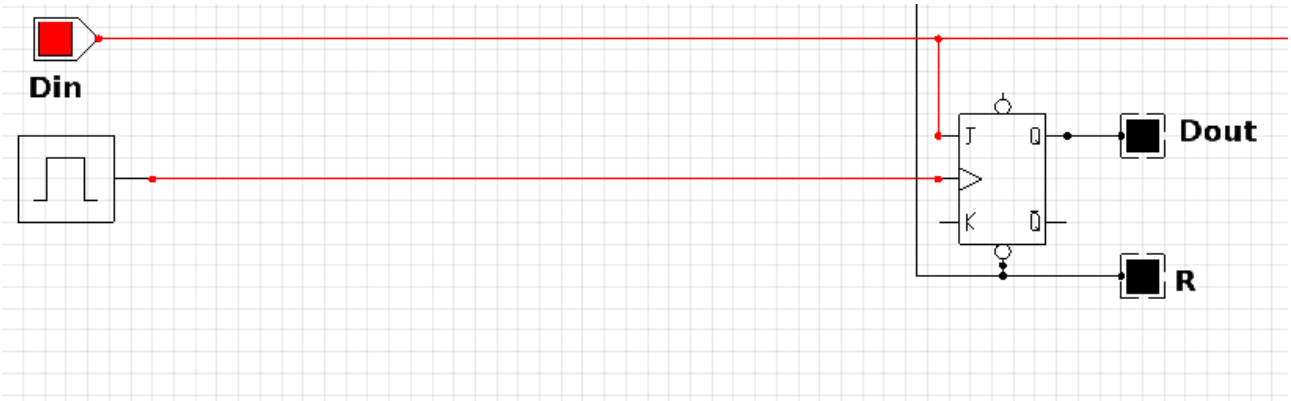
Mit dieser Technik können wir einen Zähler verwenden und die Eingänge einfach umkehren, um zwei Funktionen zu erhalten, Öffnen und Schließen, ohne dass andere komplizierte Operationen ausgeführt werden müssen

Die Motoren sind mit dem LSB (Least significant bit) verbunden, da jede Zählung ein Schritt ist, wir verbinden einfach die Schrittmotoren mit dem LBS-Signal



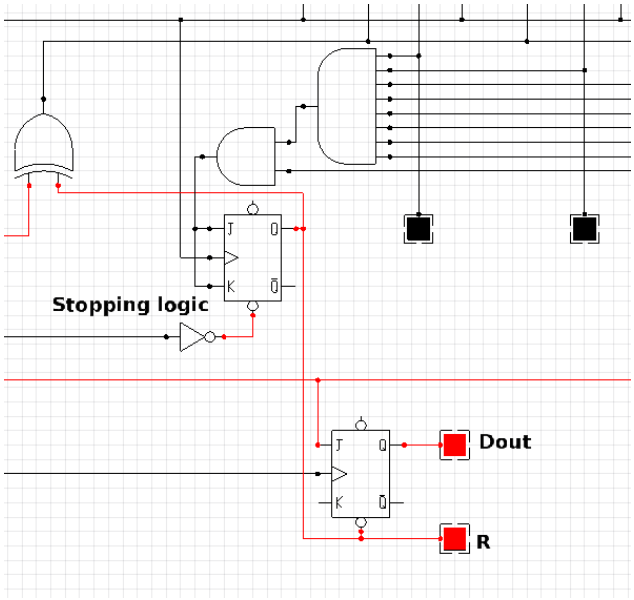


Wie bereits erwähnt, wird die Motorrichtung von Din vorgegeben, das einfach direkt an die Motoren weitergegeben wird,
Din ist auch mit einem Ausgangs-Flip-Flop verbunden, um anzuzeigen, in welchem Zustand die Türen geschlossen (0) oder offen (1) sind.



Stopplogik

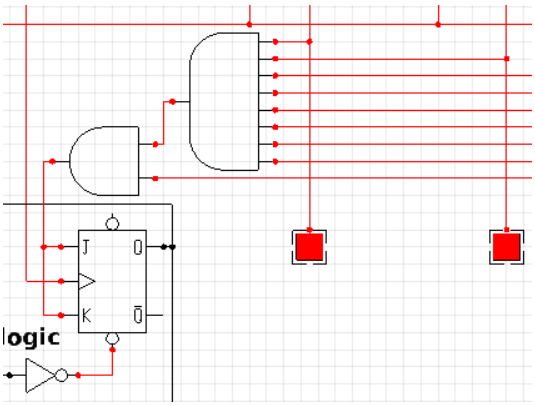
Wie der Name schon sagt, ist dieser Teil der Schaltung dafür verantwortlich, den Zähler zurückzusetzen und einen Wert an den Bus zurückzugeben, der ihm mitteilt, dass der Vorgang beendet ist und die Türen vollständig geöffnet oder geschlossen sind



Die Stopplogik selbst wird durch den C-Motoreingang aktiviert und zurückgesetzt, wodurch die Schaltung weiß, dass sie erneut zählen soll, Jede weitere Initialisierung wird durch das XOR-Gatter gestoppt, das es mit der Auslöselogik teilt, Es aktiviert auch das Flip-Flop, das für die Eingabe des Doubt-Signals verantwortlich ist.

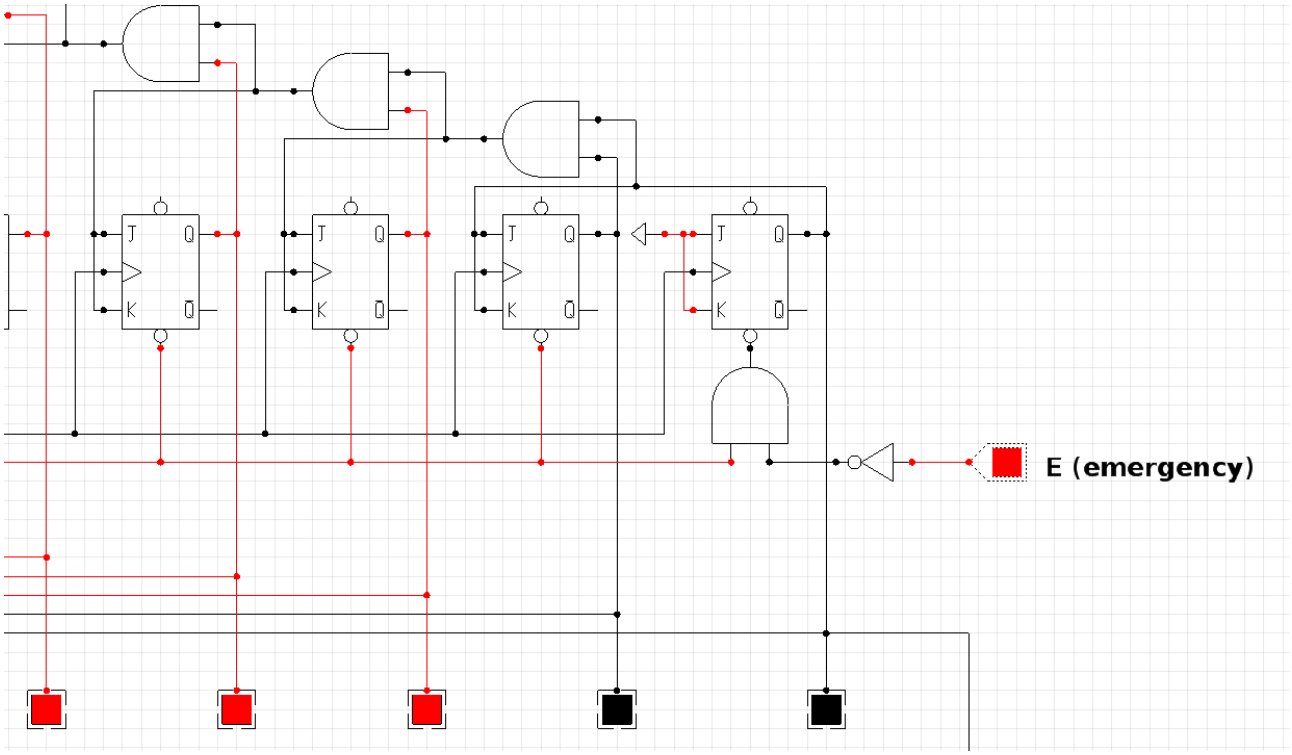
Da die Din- und C-Motorsignale mit einem Signal in der Impulsschaltung aus der letzten Aufgabe verbunden sind, wäre es logisch anzunehmen, dass sich diese beiden Signale ändern, wenn sich das ursprüngliche Signal ändert,
Theoretisch kann sich das D-Ausgangssignal ändern, wenn wir Spannung an den spezifischen Draht anlegen, aber ein solcher Fall ist unrealistisch.

Die Stopplogik ist mit einem UND-Gatter mit allen Flip-Flops im Zähler verbunden, und wenn der Zähler die maximale Zahl erreicht, empfängt das Flip-Flop ein Signal und stoppt den Zähler mit seinem Ausgangssignal



Notfall

Das Notsignal hält den Zähler davon ab, weiter fortzuschreiten, da wir unseren Zähler so eingerichtet haben, dass er sich für den Fortschritt auf das LSB verlässt, müssen wir ihn nur anhalten, um den Zähler in seinem aktuellen Zustand zu stoppen



Das Signal deaktiviert lediglich das LSB, der Negator deaktiviert es, wenn das Notsignal ausgeschaltet ist, wann das LSB ausgeschaltet ist, ist die Bedingung anderer Flip-Flops nicht erfüllt, und daher wird der Wert, den sie halten, auf unbestimmte Zeit gehalten, bis ein Reset erfolgt, wenn das Notsignal ausgeschaltet ist

Logik des Auswahlmenüs

Die Dispenser-Codierung erscheint zunächst etwas trivial, aber wir können sie für unsere Zwecke optimieren, auf den ersten Blick scheint es, dass wir 8 Codierungen benötigen, multiplizieren Sie das mit 2, da unsere Getränke entweder Eis oder Shugar enthalten können und wir übrig bleiben a 16 Kodierungen,
Auf den ersten Blick wäre es sinnvoll, einen 4x16-Bit-Codierungsmechanismus zu verwenden, aber die Vereinfachung wird später kommen.

Um uns die Mühe zu ersparen, all diese Bits zusammenzufügen, werden nur drei der vier Spalten tatsächlich zum Decodieren verwendet

0	0	0	0	Wasser
0	0	0	1	Cola
0	0	1	0	Orange
0	0	1	1	Zitrone
0	1	0	0	Tee
0	1	0	1	Kaffee
0	1	1	0	Cappucino
0	1	1	1	Latte Macchiato
1	0	0	0	Wasser mit eis
1	0	0	1	Cola mit eis
1	0	1	0	Orange mit eis
1	0	1	1	Zitrone mit eis
1	1	0	0	Tee mir Zucker
1	1	0	1	Kaffee mit Zucker
1	1	1	0	Cappucino mit Zucker
1	1	1	1	Latte Macchiato mit Zucker

Wir teilen die Codierung in kleinere separate Selektoren auf:

	Die Bits in den gelben Quadraten bestimmen, ob ein Getränk Eis / Zucker enthalten soll oder nicht ;1 mit , 0 ohne
	Die Bits, die sich in der Dekade des roten Quadrats befinden, sind das Getränk heiß oder kalt; 1 für heiß, 0 für kalt
	Die im blauen Quadrat befindlichen Bits bestimmen das tatsächliche Getränk

Auf diese Weise wird die Implementierung des Systems viel einfacher, da wir zum Beispiel Eis oder Shugar nur ausgeben müssen, wenn die Bits im gelben Kraftstoff eingeschaltet sind, was unseren Bedarf auf einen großen Decoder reduziert

Alles zusammenfügen

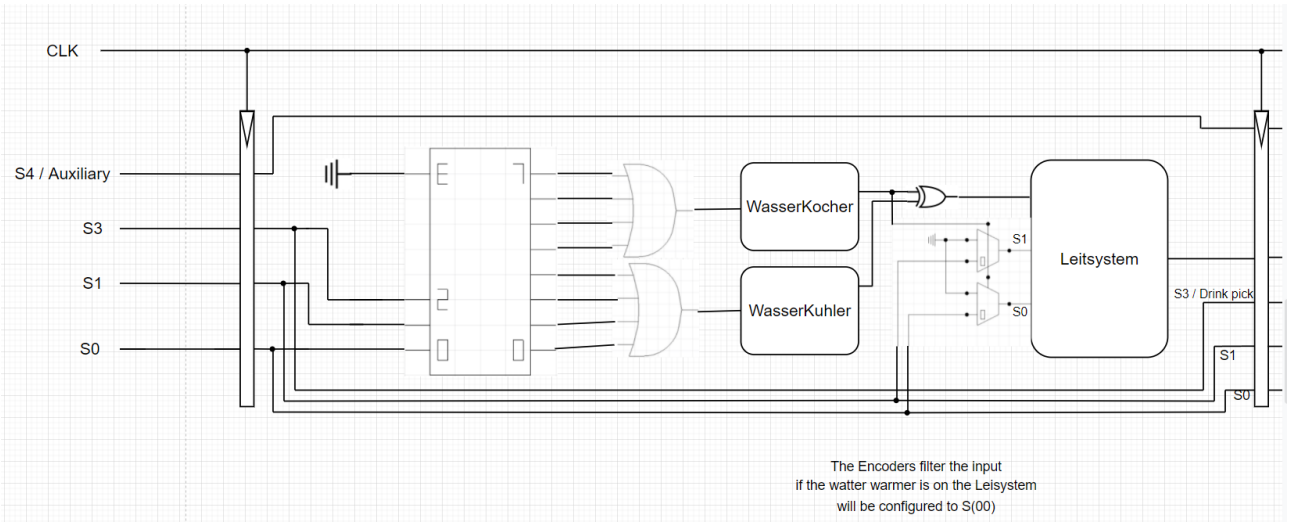
Es wird empfohlen, das Diagramm für dieses Dokument herunterzuladen, da die Screenshots von schlechter Qualität sind.

Wie zuvor besprochen, verwenden wir statt aller vier Bits nur drei Bits für den Auswahlprozess.

das System selbst gliedert sich in drei Phasen, die *Selektionsphase*, *Vorbereitungsphase* und *Dispersionsphase*

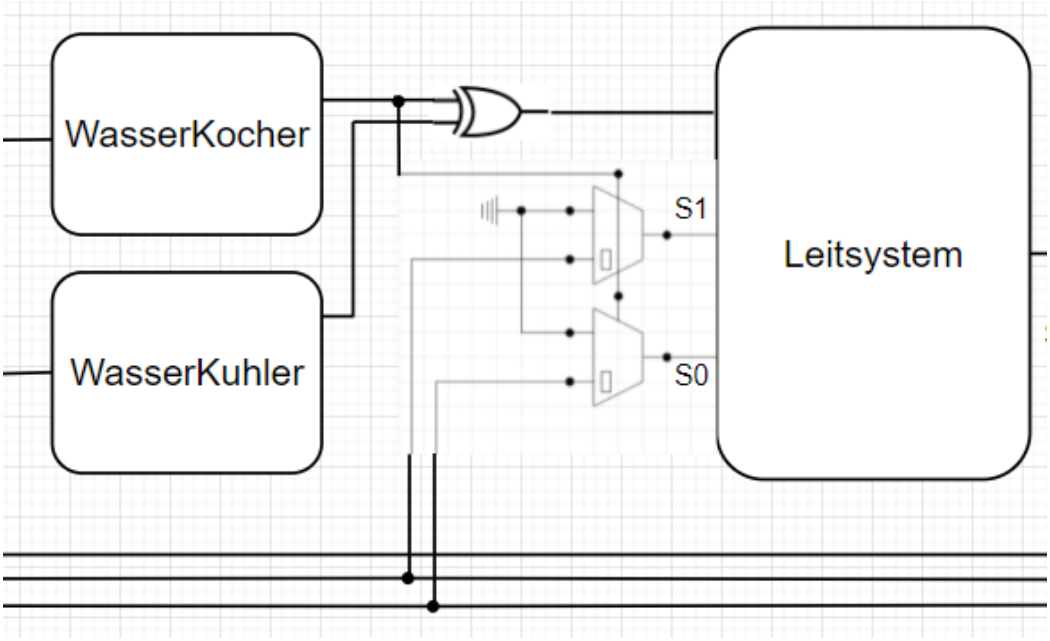
Selektionsphase

Die erste Phase besteht aus einem 3x8-Decoder, dem Wasserkühler und Wassererwärmer sowie dem Leitsystem,

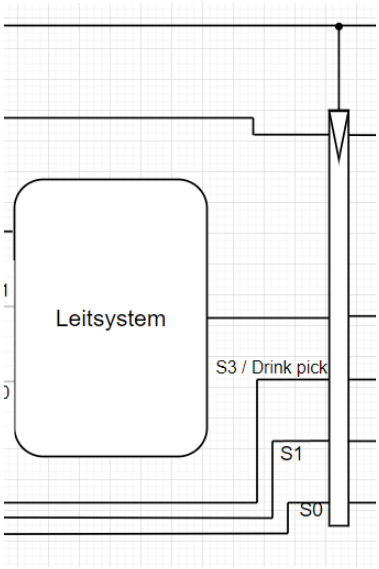


Das System nimmt die verschlüsselten Eingaben eines Getränks, führt es durch den Decoder und aktiviert entweder den Kühler oder den Wärmer.

das Leitsystem wird dann basierend auf der Eingabe sowie der gewählten Funktion konfiguriert,



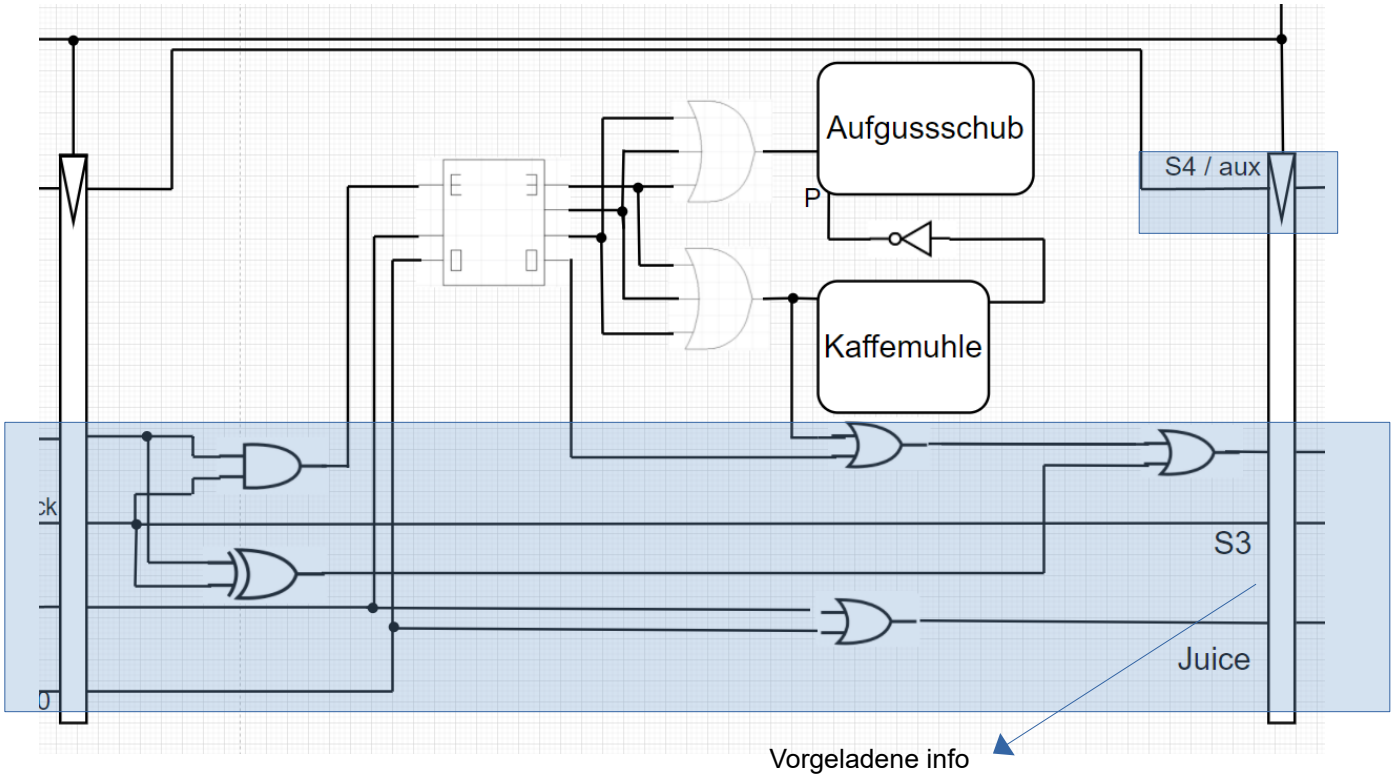
Die Auswahlschaltung besteht aus den Multiplexern, die Multiplexer nehmen das Signal von S0 und S1 auf, die Bits, die bestimmen, für welches Getränk das System konfiguriert werden soll,
wenn der Wasserkühler aktiv ist, akzeptiert das Leitsystem eine Kombination aus S0 und S1, wenn der Wasserwärmer eingeschaltet ist, wird das Leitsystem immer auf Wasser konfiguriert (S0 = 0 , S1 = 0), Es würde keinen Sinn machen, Wasser aufzuwärmen und dann Sirup auszuwählen, der anstelle von Tee ausgegeben wird



schließlich geben wir die Eingänge an die nächste Stufe weiter, *S3, S1, S0* sowie *St* das Initialisierungssignal von Ihrem Leitsystem

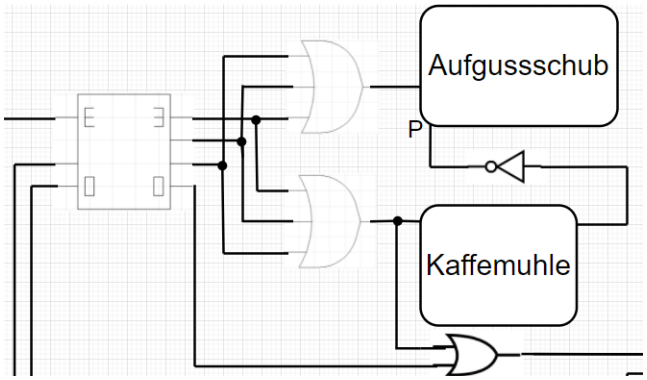
Vorbereitungsphase

In dieser Phase werden die gesamten Vorbereitungen für die Abgabephase durchgeführt, wobei ein Decoder verwendet wird, um genau auszuwählen, welches Getränk zubereitet werden soll, und die erforderlichen Aktionen ausgeführt werden, sei es das Mahlen von Kaffee oder das Weitergeben der Eingaben an die nächste Phase.



Je nach Stufe werden die Informationen der vorherigen Stufe an die nächste Stufe weitergegeben, während die nächste Stufe noch inaktiv ist, wodurch die Informationen effektiv vorgeladen werden,
Je nach Getränk, ob es sich nur um Saft oder Tee handelt, wird die nächste Stufe so bald wie möglich initialisiert, wenn Kaffee oder spät ausgewählt ist, müssen die Zubereitungen beginnen.

Die Zubereitung erfolgt nach Rezept, gesteuert durch den Decoder.

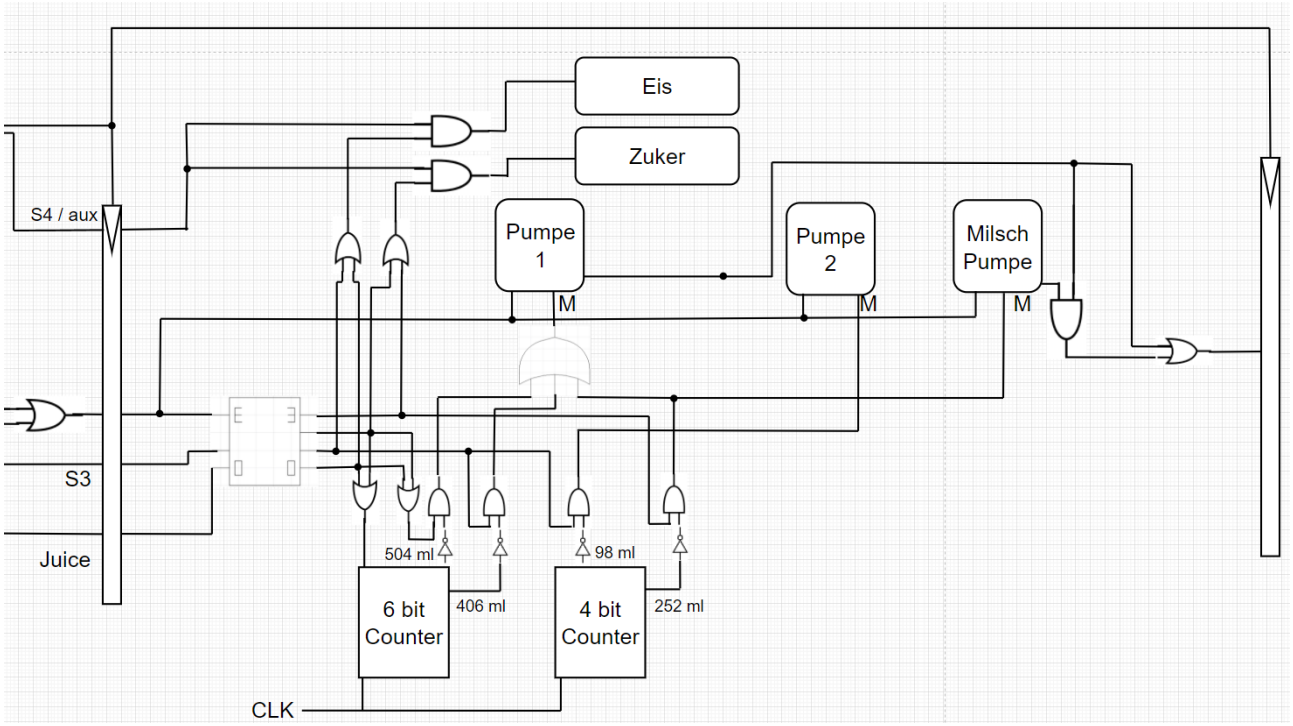


Nur die erste Eingabe führt zur nächsten Stufe, ohne andere Funktionen auszulösen, dh alle anderen Kombinationen lösen beide Funktionen aus.

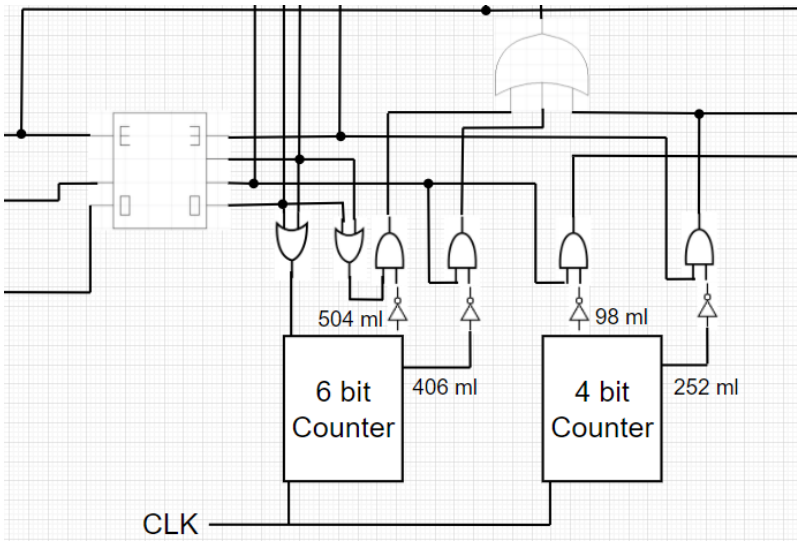
0	0	Tee
0	1	Kaffe
1	0	Cappucino
1	1	Late

Dispersionsphase

Hier geben wir aus, dieser Teil besteht aus drei Pumpen (2 normale und 1 Milchpumpe), einem 2x4-Decoder, zwei Zählern, Eis- und Zuckerspender,



Das Highlight dieses Teils ist der Decoder, per Anleitung hätten wir hier wieder das Leitsystem verwenden können, aber mit einer geschickten Implementierung des Decoders können wir die Notwendigkeit dafür vollständig negieren,
der Decoder nimmt den Übertrag von den letzten zwei Stufen auf



Da wir alle Vorbereitungen getroffen haben, um unser Getränk auszugeben, müssen wir nur auswählen, welcher Zähler initialisiert werden soll. Die Zähler selbst sind die gleichen Zähler, die in Aufgabe 2.1 untersucht wurden, außer mit nicht so vielen Bits. Erinnern Sie sich, wie ti funktioniert, wir Hace und Gates verbinden der Zähler geworfen, dies ermöglicht uns nun, einzelne Drähte aus einzelnen Bits zu ziehen,

Im Wesentlichen müssen wir die Zählung weiterzählen, bis wir unser bestimmtes Bit treffen;

$log_2(500/14) = 5.158… \sim 6$ bits

500/14 = 35.714.. ~ 36 steps

*36 * 14 = 504 ml*

$log_2(400/14) = 4.836… \sim 5$ bits

400/14 = 28.571… ~ 29 steps

*29 * 14 = 406ml*

$log_2(100/14) = 2.836… \sim 3$ bits

100/14 = 7.142… ~ 7 steps

*7 * 14 = 98*

$log_2(250/18) = 3.798… \sim 4$ bits

250/18 = 13.888… ~ 14 steps

14 18 = 252 ml*

Diese mathematische Gleichung gibt uns alle Variablen, die wir brauchen, hauptsächlich die Zahl, die unser Zähler erreichen muss, um die höchstmögliche Zahl zu erreichen. Der Nachteil ist, dass wir nicht genau sein können, aber wir können dieselben Zähler wiederverwenden, außer mit unterschiedlichen Zahlensiegel.

now all we need to do is keep the signal to the pumps active until our counters reach the desired step number, Dies wird durch die damit verbundenen Negative noch verstärkt. Um uns selbst Komponenten zu sparen, prüfen Sie nur, ob unser Zähler das gewünschte Bit erreicht hat, an dem der Ausgang wahr ist und das Signal der Pumpen stoppt.

die Dekodierungstabelle;

0	0	wasser
0	1	cola/orange/zitron
1	0	tee/kaffe
1	1	cappucino/latte

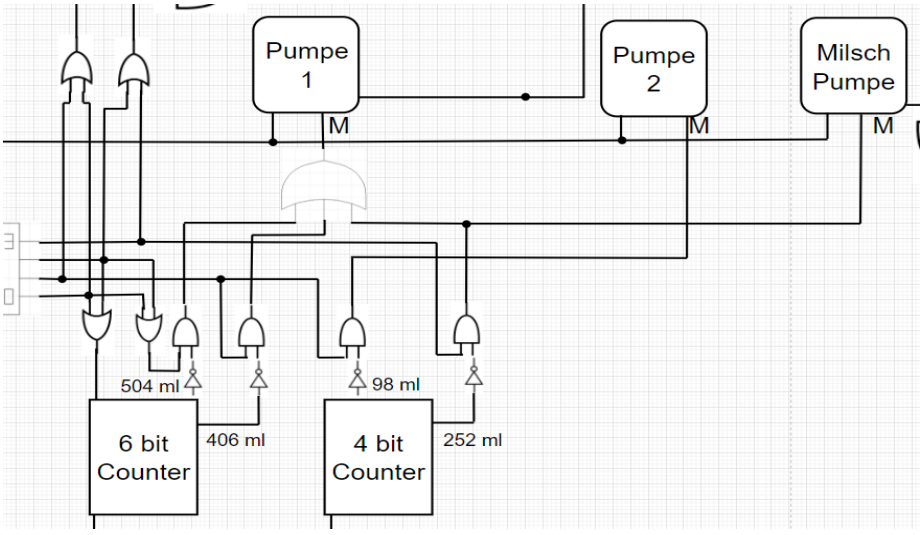
da wir die Zähler wiederverwenden und gleichzeitig ausführen können. Alles, was wir tun müssen, ist auszuwählen, welche Zählung aktiv sein soll und für wie viele Schritte, da der Decoder zwei Eingänge S3 (die das Getränk heiß oder kalt auswählen) und Juice (S0 + S1) hat, können wir auf die Getränkerezepte verweisen , können wir sie in vier Signale unterteilen,

1. Wasser die den ersten Zähler mit 504-ml-Konfigurationsschalter aktivieren

2. Säfte (Cola/Orange/Zitron) schalten die erste Theke mit 406 ml Konfiguration ein und und der zweite Zähler mit 90-ml-Konfiguration

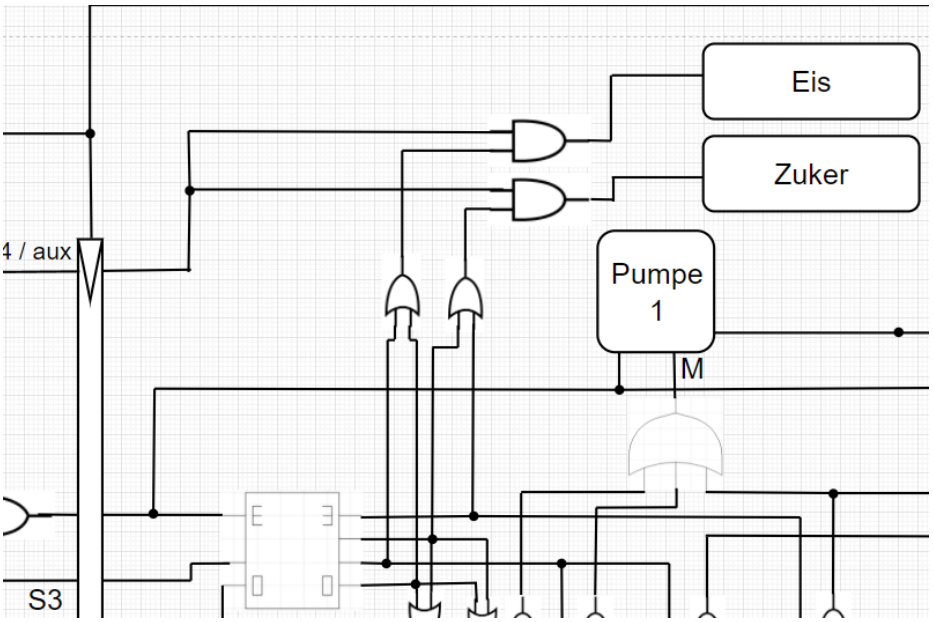
3. Tee und Kaffee die den ersten Zähler mit 504-ml-Konfigurationsschalter aktivieren

4. Cappuccino und Latte, die auf der zweiten Theke mit 250-ml-Konfiguration aktiviert werden



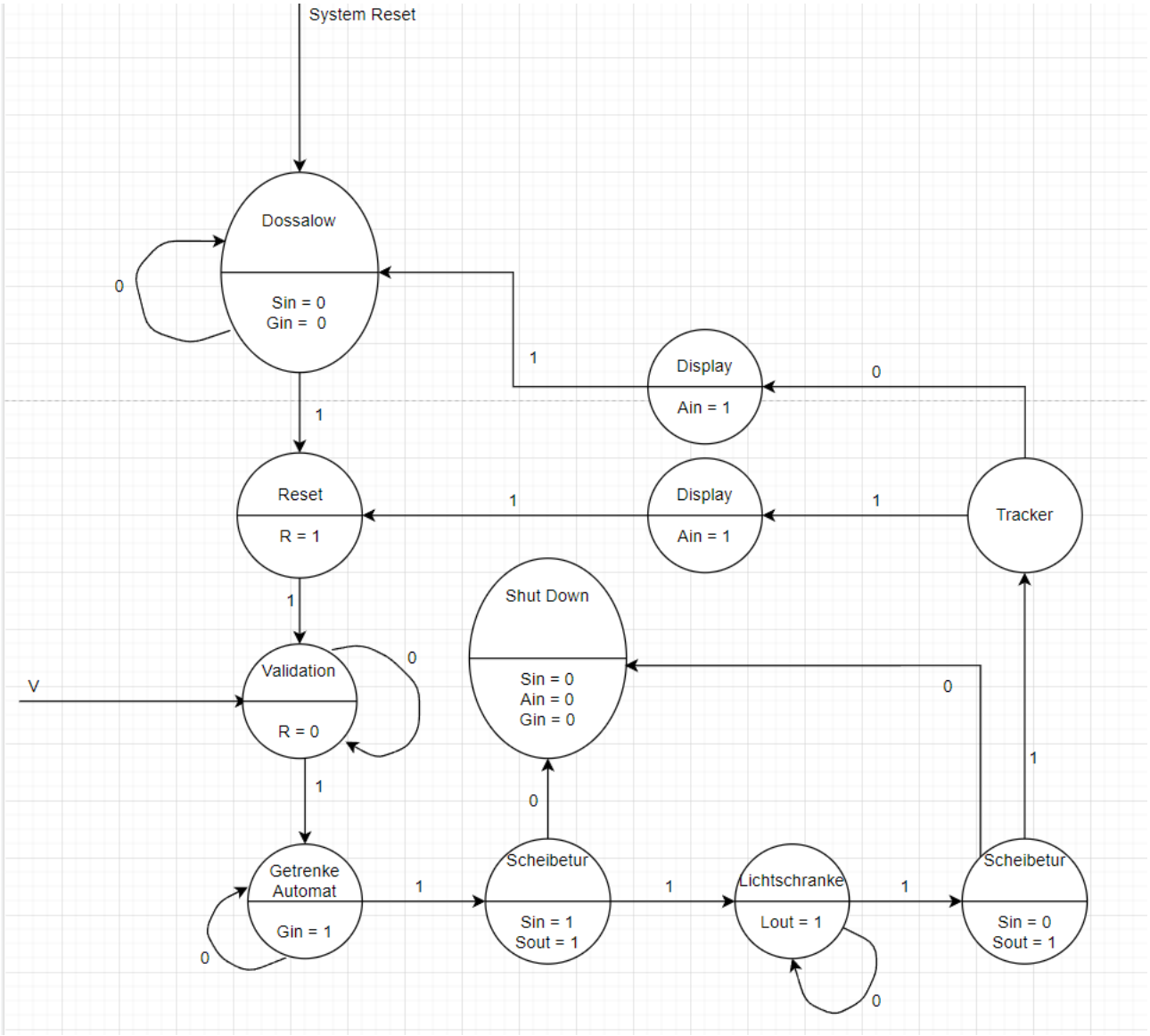
Jede Zählerkonfiguration ist mit ihrer entsprechenden Pumpe verbunden, Die Pumpen laufen parallel, deshalb brauchen wir zwei normale Pumpen, eine für die Ausgabe von Wasser und die andere für Sirup für Säfte

Abschließend wird dabei je nach Eingang von S4 und Decoder das Eis oder der Zucker ausgegeben, Dieser Teil des Systems hängt vom Decoder ab, sodass vor seiner Aktivierung keine Arbeiten durchgeführt werden, um sicherzustellen, dass Eis und Zucker zuletzt ausgegeben werden



Maschinenbetriebsdiagramm

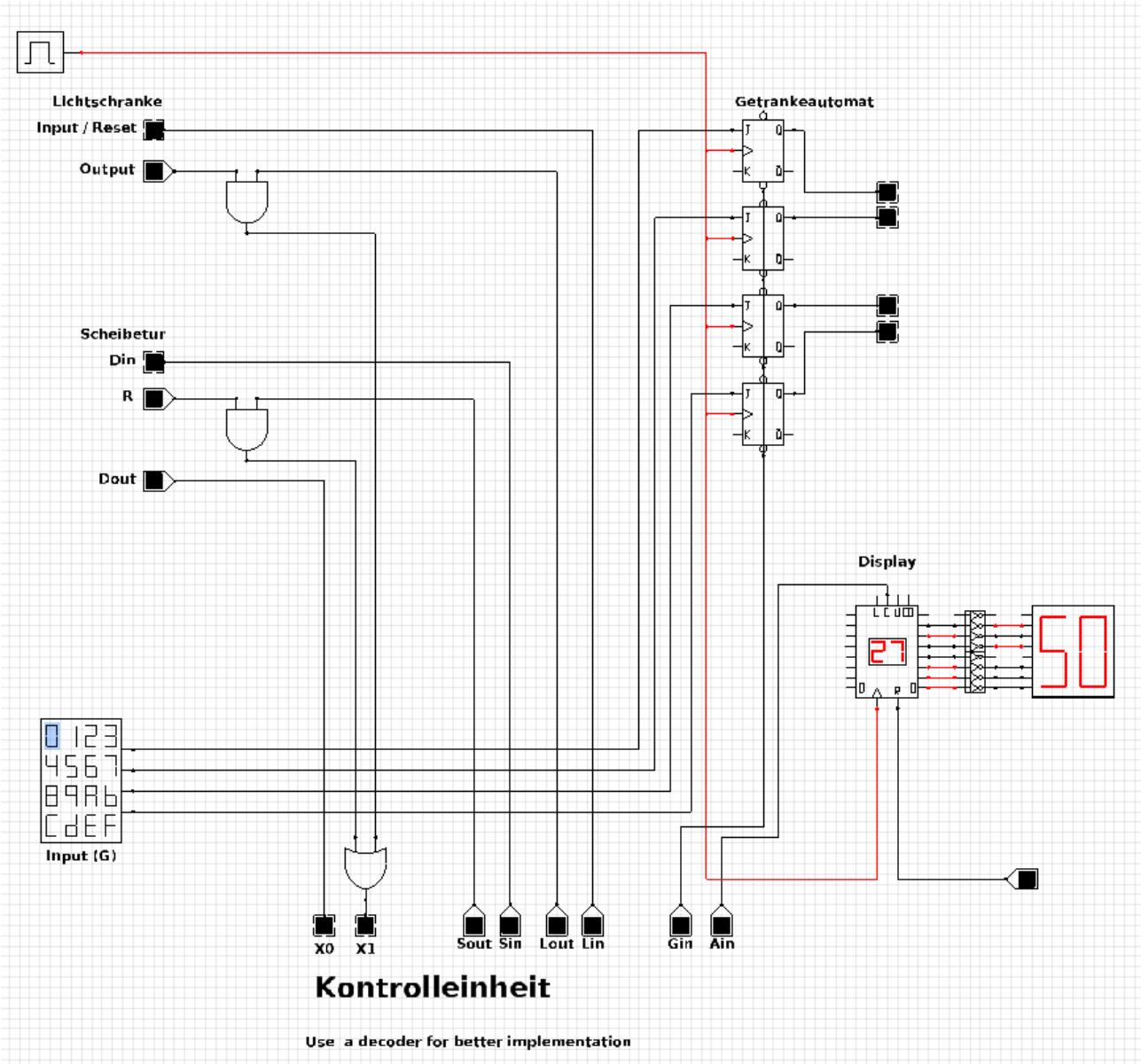
Die endliche Zustandsmaschine zeigt die Steuerschaltung und was jeder Zustand steuert, sowie seine Ausgabe an den Bus



Die Maschine verfügt über zwei zusätzliche Zustände, einen Diallow-Zustand, in den eingetreten wird, wenn die Anzahl der unterstützten Besucher überschritten wird und verhindert, dass jemand anderes eintreten kann, und einen Tracker, der die Anzahl der anwesenden Kunden verfolgt.

Bussystem und Display-Controller

Der Bus verbindet alle Komponenten miteinander



Die Schaltung kann mit Decoder besser gemacht werden, da wir 4 Module haben und alles, was wir brauchen, zwei Eingänge sind, mit einem 2x4-Decoder können wir die Anzahl der erforderlichen Steuerbits auf 2 reduzieren