

DT-Projekt 2

2.1 Schiebetür (8 PP)

In dieser Aufgabe sollen Sie digitale Schaltungen für die Schiebetür und die zugehörige Lichtschranke designen, und dafür in jeder Teilaufgabe einen Schaltplan zeichnen. Beides sind getrennte Komponenten, die jeweils an den Bus angeschlossen sind. Die Schiebetür wird mit zwei Schrittmotoren betrieben, einen für die rechte und einen für die linke Tür. Sobald ein Ticket gescannt und vom Kontrollsysteem als valide identifiziert wird, soll sich die Schiebetür öffnen. Sobald die Lichtschranke unterbrochen wird, soll die Tür wieder geschlossen werden.

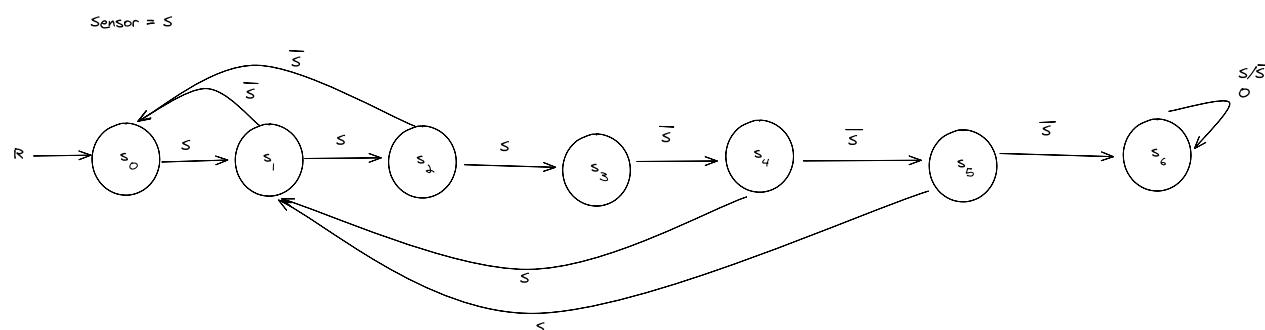
Hinweis: Sie müssen an dieser Stelle nicht die Kommunikation mit dem Bus beachten, d.h., Sie dürfen davon ausgehen, dass die Eingänge stets valide sind.

a)

Entwerfen Sie zunächst eine digitale Schaltung für die Kontrolleinheit der Lichtschranke. Diese enthält den Eingang Sensor, der direkt an den Sensor angeschlossen ist. Der Sensor gibt eine logische 1 aus, wenn die Lichtschranke unterbrochen ist und eine 0, wenn dies nicht der Fall ist. Die Kontrolleinheit der Lichtschranke hat einen Ausgang zum Bus, den wir mit O gekennzeichnet haben, und einen Eingang vom Bus, den wir mit R bezeichnen. Am Ausgang O soll eine logische 1 anliegen, sobald ein gültiges Passieren eines Kunden erkannt wird. Ein gültiges Passieren findet genau dann statt, wenn die Lichtschranke für mindestens 3 Takte unterbrochen ist und anschließend für 3 Takte ununterbrochen bleibt.

Falls das Lichtschranken-Kontrollsysteem ein gültiges Passieren entdeckt, soll solange am Ausgang O die 1 ausgegeben werden, bis ein Reset Signal am Eingang R anliegt. (2 PP)

Um die Kontrolleinheit zu bauen, können wir zuerst einen Mealy-Automaten erstellen, der die Lichtschranke darstellt. Diesen können wir dann in seine Zustands- und Ausgabetabellen verwandeln und diese mithilfe von Espresso minimieren.



Zustand	Sensor	Nächster Zustand
S_0	0	s_0
S_0	1	s_1
S_1	0	s_0
S_1	1	s_2
S_2	0	s_0
S_2	1	s_3
S_3	0	s_4
S_3	1	s_3
S_4	0	s_5
S_4	1	s_1
S_5	0	s_6
S_5	1	s_1
S_6	-	s_6

Zustand	s_2	s_1	s_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	0
S_3	0	1	1
S_4	1	0	0
S_5	1	0	1
S_6	1	1	0

Zustand	Sensor	0
S_0	-	0
S_1	-	0
S_2	-	0
S_3	-	0
S_4	-	0
S_5	-	0
S_6	-	1

Als input für die Zustände haben wir:

```
.i 4
.o 3
0000 000
0001 001
0010 000
0011 010
0100 000
0101 011
0110 100
0111 011
1000 101
1001 001
1010 110
1011 001
110- 110
111- ---
```

Damit generiert Espresso:

```
.i 4
.o 3
.p 8
11-- 110
1-10 110
0-11 010
-110 100
01-1 011
1000 101
10-1 001
-001 001
.e
```

Daraus erhalten wir:

$$S'_2 = s_2 s_1 + s_2 s_0 \bar{S} + s_1 s_0 \bar{S} + s_2 \bar{s}_1 \bar{s}_0 \bar{S}$$

$$S'_1 = s_2 s_1 + s_2 s_0 \bar{S} + \bar{s}_2 s_0 S + \bar{s}_2 s_1 S$$

$$S'_0 = \bar{s}_2 s_1 S + s_2 \bar{s}_1 \bar{s}_0 \bar{S} + s_2 \bar{s}_1 S + \bar{s}_1 \bar{s}_0 S$$

Für die Outputs haben wir: (Wobei hierfür Espresso ziemlich overkill ist, da man sofort sieht, dass 0 exakt dann 1 ausgibt, wenn wir in Zustand S_6 sind)

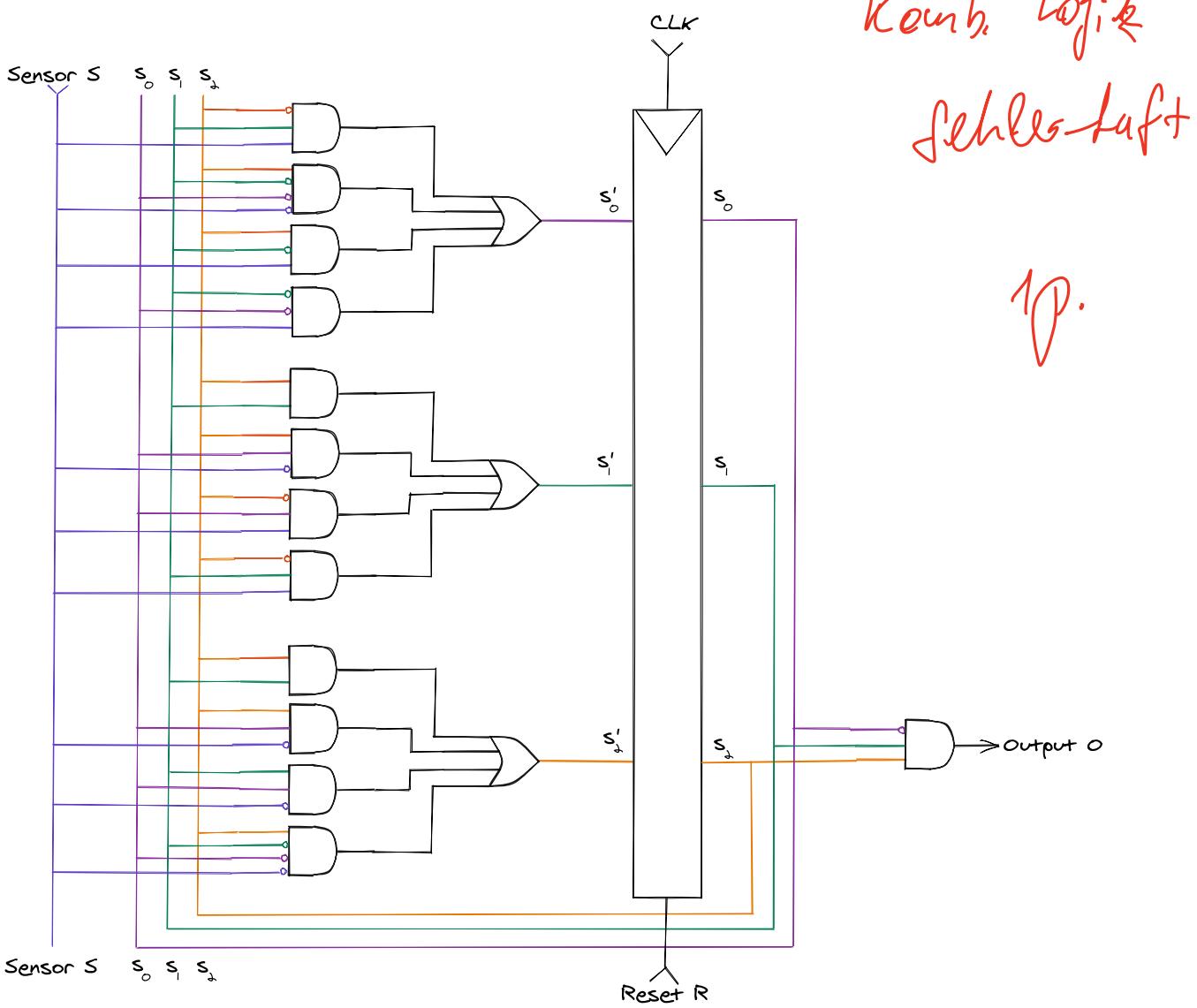
```
.i 4
.o 1
000- 0
001- 0
010- 0
011- 0
100- 0
101- 0
110- 1
```

```
.i 4
.o 1
.p 1
110- 1
.e
```

Das heißt:

$$O = s_2 s_1 \overline{s_0}$$

Zu guter letzt müssen wir damit nur noch einen Schaltplan entwerfen.



b)

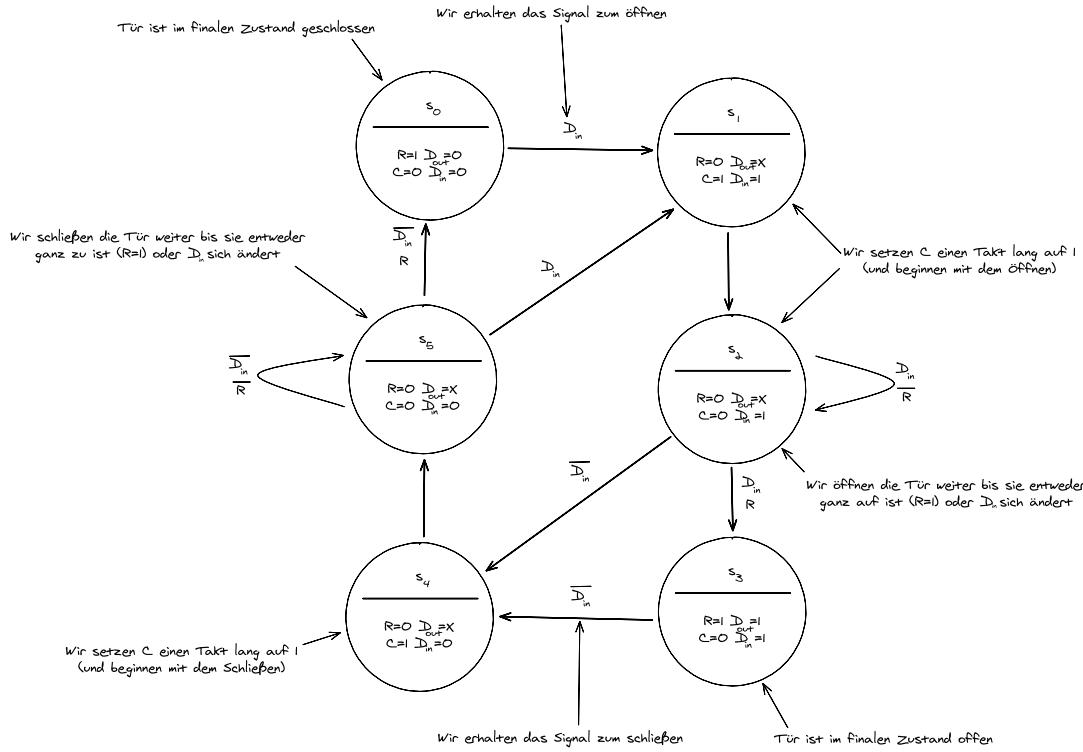
Entwerfen Sie eine digitale Schaltung für die Schiebetür, die folgende Spezifikationen erfüllt. Die Schaltung erhält den Eingang D_{in} vom Bus, der angibt, ob die Tür geöffnet ($D_{in} = 1$) oder geschlossen werden soll ($D_{in} = 0$). Des Weiteren hat die Schiebetür 2 Ausgänge zum Bus, die wir mit R und D_{out} bezeichnen. R ist auf 1 gesetzt, wenn die Schiebetür eine finale Position erreicht hat, d.h. komplett geöffnet oder geschlossen ist. D_{out} hingegen gibt an, in welcher finalen Position sich die Schiebetür befindet (1 = offen, 0 = geschlossen).

Für diesen Teil der Schiebetür haben Sie zunächst Zugriff auf einen Motorcontrol-Chipbaustein (vgl. Abbildung 1), der sich um die Kontrolle der Schrittmotoren kümmert. Dieser erhält die beiden Eingänge D_{in} und C , sowie

die beiden Ausgänge R und D_{out} . Die Ausgänge haben dieselbe Funktion wie die gleichnamigen Ausgänge der kompletten Schiebetür. Wie zuvor gibt der D_{in} -Eingang an, ob die Tür geöffnet (1) oder geschlossen (0) werden soll.

Der C -Eingang wird auf 1 gesetzt, wenn die Schiebetür die am Eingang D_{in} angegebene Richtung anfahren soll. C soll dabei nur für einen Takt auf 1 gesetzt werden und auch nur, wenn sich die Richtungsanweisung, die vom Bus-Eingang D_{in} kommt, ändert. Beachten Sie, dass sich die Richtung während der Bewegung ebenfalls ändern kann. (1 PP)

Wir fangen wieder an, indem wir einen Automaten entwerfen, allerdings dieses mal einen Moore-Automaten, da sich dieser eher anbietet:



Der Output D_{out} vom Motorcontrol-Bauteil ist redundant, da wir den D_{out} Output von dem Bauteil Schiebetür mithilfe von D_{in} und R erhalten können. Hier könnte man einen Vergleich einbauen, der bei Ungleichheit einen Fehler ausgibt.

Da bei nicht-Endzuständen (also nicht s_0 oder s_3) nicht klar und auch nicht wichtig ist welchen Wert D_{out} hat, können wir diesen mit Don't Cares markieren.

Wie bei a) machen wir auch hier weiter, indem wir die Zustände kodieren und eine Zustandsübergangstabelle und Ausgabetafelle aufstellen:

Zustand	D_{in}	R	Nächster Zustand
S_0	0	-	s_0
S_0	1	-	s_1
S_1	-	-	s_2
S_2	0	-	s_4
S_2	1	0	s_2
S_2	1	1	s_3
S_3	0	-	s_4
S_3	1	-	s_3
S_4	-	-	s_5
S_5	0	0	s_5
S_5	0	1	s_0
S_5	1	-	s_1

Zustand	s_2	s_1	s_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	0
S_3	0	1	1
S_4	1	0	0
S_5	1	0	1

Zustand	R	D_{out}	C	D_{in}
S_0	1	0	0	0
S_1	0	-	1	1
S_2	0	-	0	1
S_3	1	1	0	1
S_4	0	-	1	0
S_5	0	-	0	0

Jetzt können wir wieder Espresso benutzen, um die Tabellen zu minimieren und um eine bool'sche Gleichung aufzustellen:

Für die Inputs:

```
.i 5
.o 3
0000- 000
0001- 001
001-- 010
0100- 100
01010 010
01011 011
0110- 100
0111- 011
100-- 101
10100 101
10101 000
1011- 001
11--- ---
```

Espresso-Output für die Inputs:

```
.i 5
.o 3
.p 9
001-- 010
--011 001
1--00 101
-001- 001
-111- 001
-1-0- 100
-1-1- 010
1--1- 001
1-0-- 101
.e
```

Das heißt:

$$S'_2 = s_2 \overline{D_{in}} \overline{R} + s_1 \overline{D_{in}} + s_2 \overline{s_0}$$

$$S'_1 = \overline{s_2 s_1} s_0 + s_1 D_{in}$$

$$S'_0 = \overline{s_0} D_{in} R + s_2 \overline{D_{in}} \overline{R} + \overline{s_1 s_0} D_{in} + s_1 s_0 D_{in} + s_2 D_{in} + s_2 \overline{s_0}$$

Für die Outputs:

```
.i 3
.o 4
000 1000
001 0-11
010 0-01
011 1101
100 0-10
101 0-00
11- ----
```

Espresso-Output für die Outputs:

```
.i 3
.o 4
.p 5
000 1000
1-0 0010
001 0011
-11 1000
-1- 0101
.e
```

Das heißt:

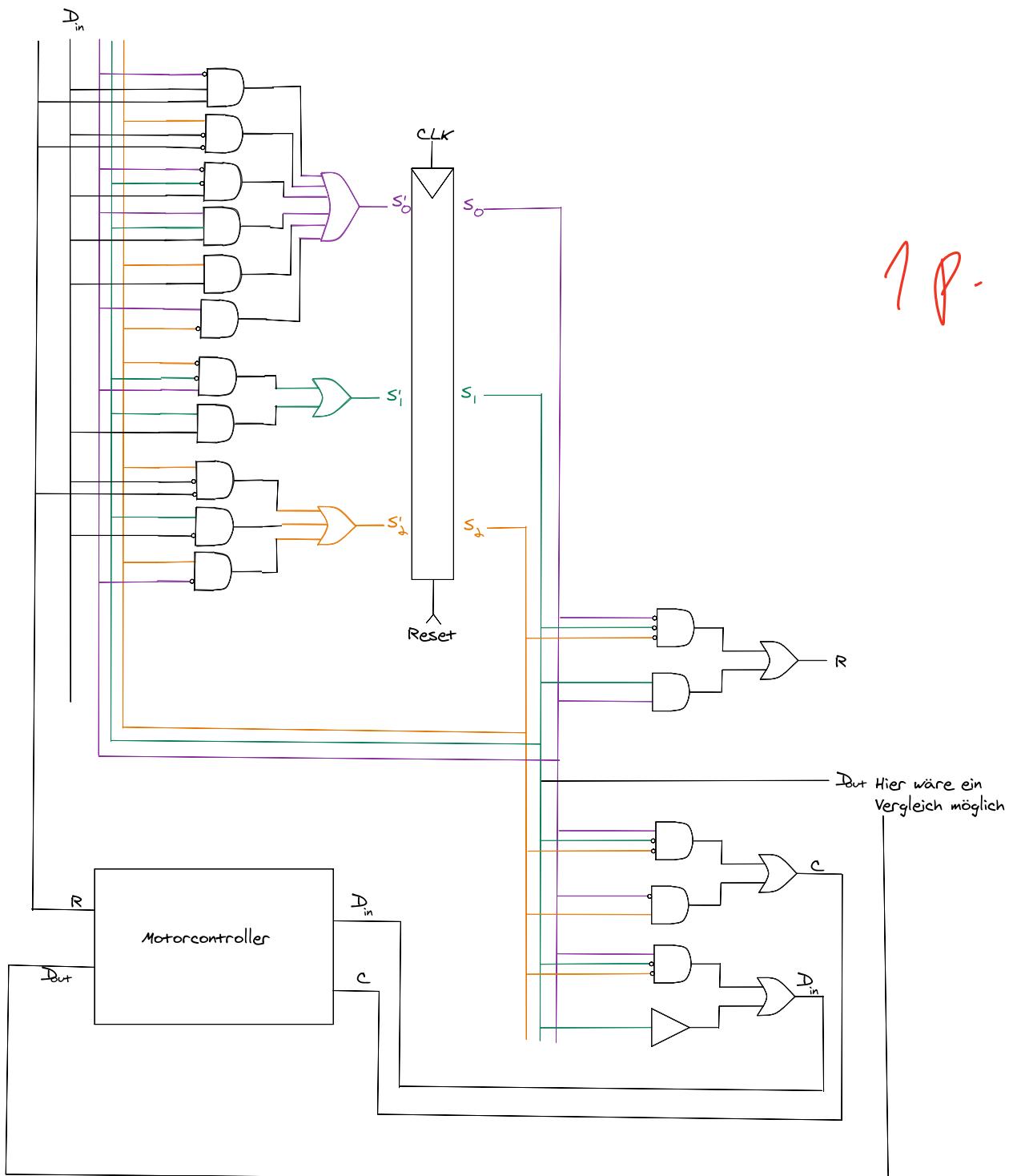
$$R = \overline{s_2 s_1 s_0} + s_1 s_0$$

$$D_{out} = s_1$$

$$C = s_2 \overline{s_0} + \overline{s_2 s_1} s_0$$

$$D_{in} = \overline{s_2 s_1} s_0 + s_1$$

Damit erhalten wir folgende digitale Schaltung:

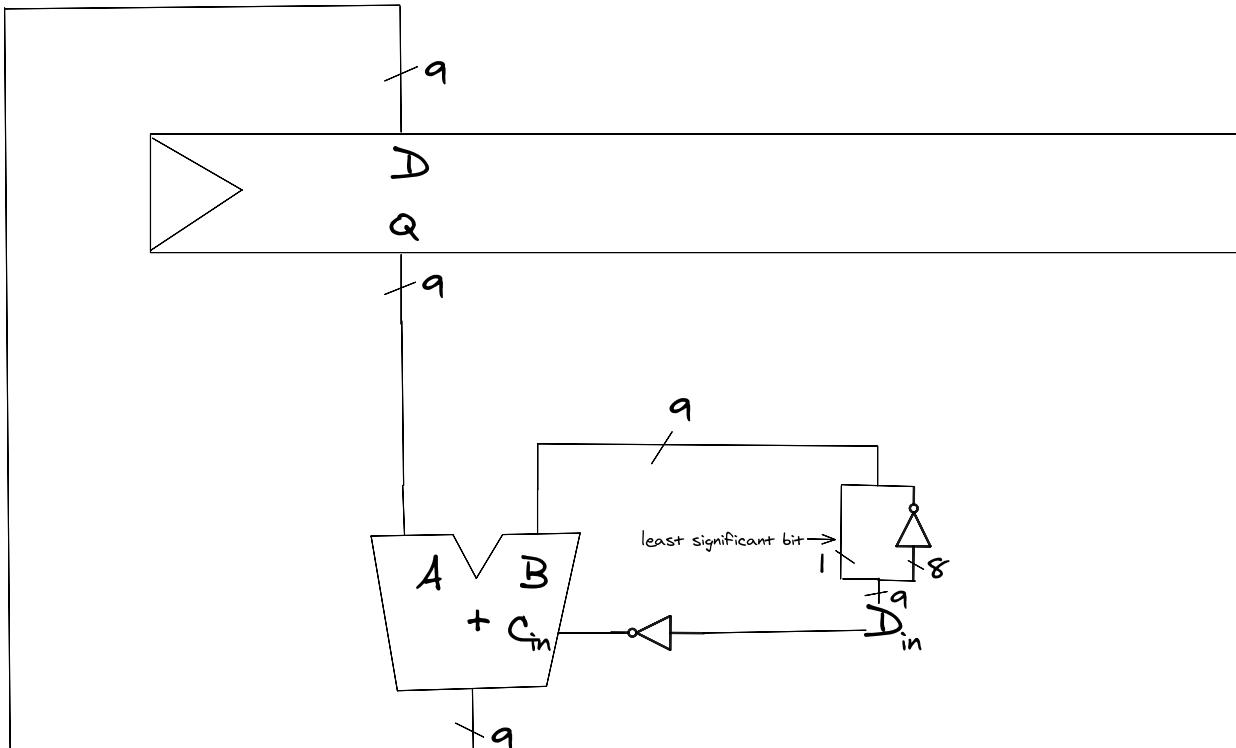


c)

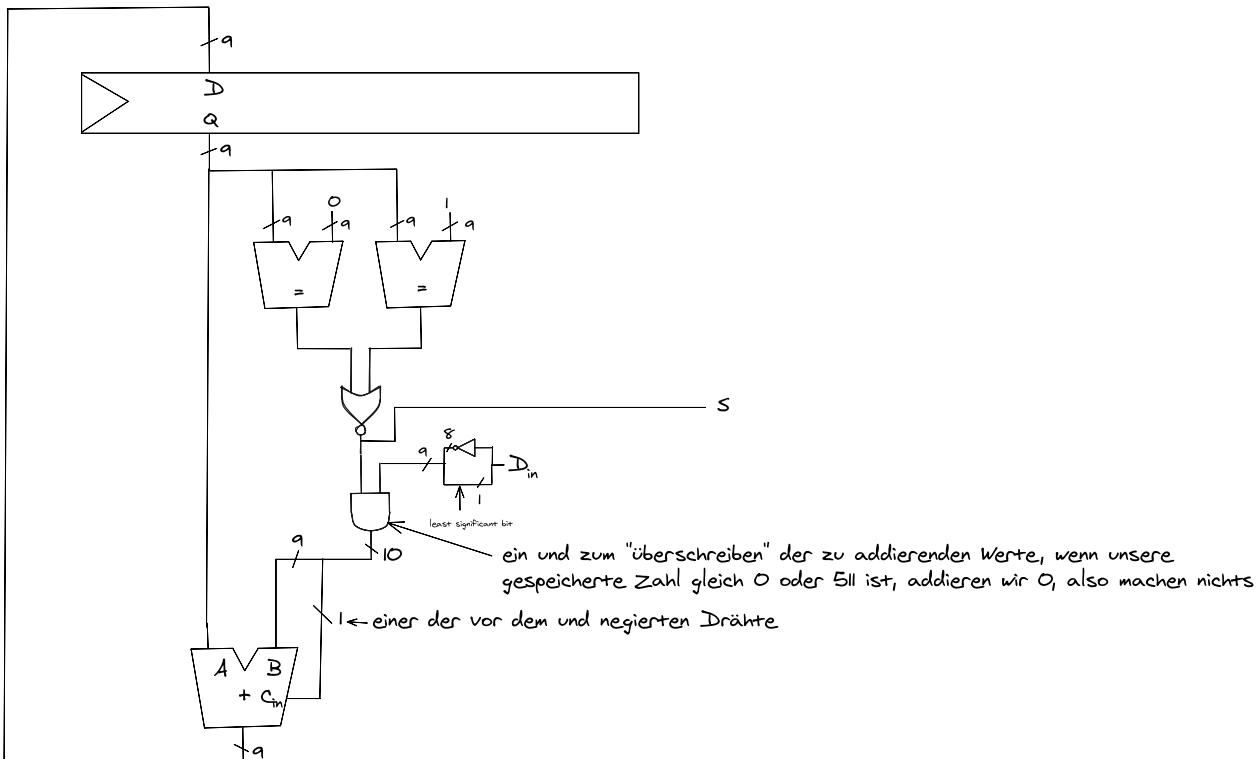
Abschließend sollen Sie eine digitale Schaltung für den Motorcontrol-Chipbaustein entwerfen, den Sie in Ihrer Schaltung aus Teilaufgabe b) verwendet haben. Dieser Chipbaustein soll die Motoren steuern, um die Tür zu öffnen und zu schließen. Verwenden Sie dazu einen Motor-Chipbaustein, der die beiden Eingänge S und D_{in} hat (vgl. Abbildung 1). Der Schrittmotor hat insgesamt 512 Positionen, wobei die Tür sich bei Position 0 im geschlossenen und bei Position 511 im offenen Zustand befindet (für beide Türen). Wenn der Eingang S des Motor-Chipbausteins 1 ist, dann bewegt sich der Motor innerhalb eines Taktes in die an D_{in} angegebene Richtung. Bezeichnen Sie den Motor-Chipbaustein für die linke Tür mit $Motor_L$ und den für die rechte Tür mit $Motor_R$. Achten Sie darauf, dass der Motor

niemals über die angegeben Schrittstufen hinaus angesteuert wird. Sie können davon ausgehen, dass sich die Motoren initial an Position 0 (Tür geschlossen) befinden. Zusätzlich zu den genannten Eingängen hat der Motorcontrol-Chipbaustein einen EMERGENCY-Eingang E, der bei einem Notfall auf 1 gesetzt ist und die Tür sofort anhält. Der Betrieb geht erst weiter, sobald E wieder auf 0 gesetzt ist. (5 PP)

Zuerst brauchen wir eine Schaltung, die abhängig von D_{in} hoch bzw runterzählt, dies können wir mit 9 D-FF und 9 Full-Addern implementieren. Unser D_{in} muss dafür entweder 00000001 sein und wir haben kein extra Übertrag, wenn wir 1 zu unserer gespeicherten Zahl in den D-FF addieren möchten und wenn D_{in} 11111110 ist und wir einen Übertrag von 1 haben, subtrahieren wir 1 von dieser Zahl.

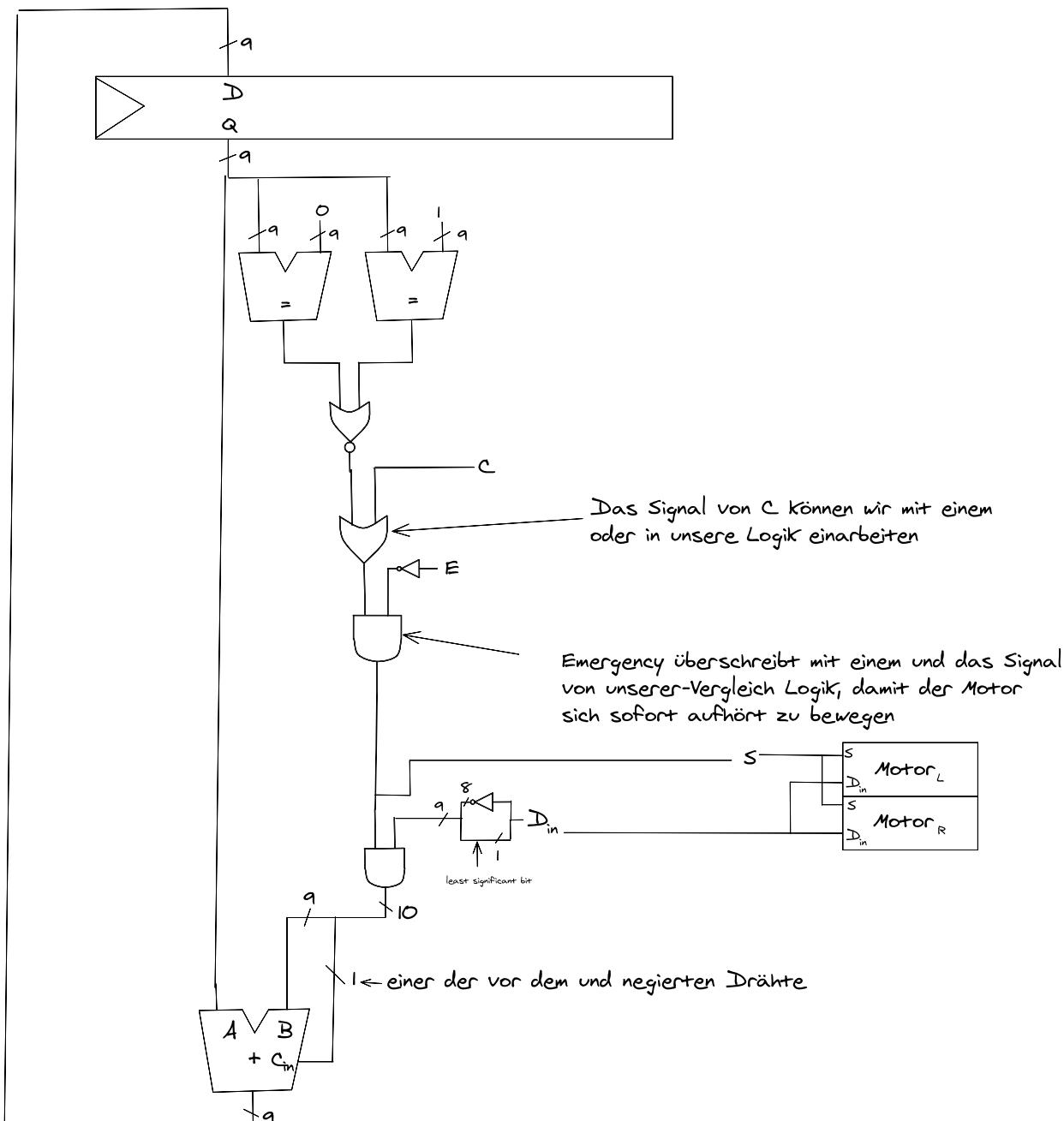


Jetzt fehlt noch die Logik, dass wir nur dann die Zahl ändern und dem Motor ein S-Signal geben, wenn die Zahl zwischen 1 und 510 ist (also nicht 0 oder 511).



Jetzt fehlt noch das "anstoßen" des Motors, also dass wir den Wert ändern und S auf 1 setzen, sobald wir C erhalten (einfach mit einem oder umzusetzen) und wir dürfen unser Notaus E nicht vergessen (ein weiteres und, um den Wert zu überschreiben):

Und die Motor-Bausteine $Motor_L$ und $Motor_R$ die identisch sind.



SP,

Getränk	Rezept
Wasser	(Wasserkühler + Leitsystem) → Pumpe (500ml)
Cola/Orange/Zitrone	(Wasserkühler + Leitsystem) → Pumpe (400ml) → Leitsystem → Pumpe (100ml)
Tee	(Wasserkocher + Leitsystem) → Pumpe (500ml)
Kaffee	Wasserkocher + Leitsystem + (Kaffeemühle → Aufgusschub ($P = 1$)) → Pumpe (500ml) → Aufgusschub ($P = 0$)
Cappuccino / Latte Macchiato	Wasserkocher + Leitsystem + (Kaffeemühle → Aufgusschub ($P = 1$)) → Pumpe (250 ml) → Aufgusschub ($P = 0$) → Milchpumpe (250 ml)

Tabelle 1: Rezepte der Getränke des Getränkeautomaten. Jedes Rezept beginnt mit dem Becherauslass sowie der Nutzung des Leitsystems vom Wassertank zum Auslass (dies kann auch parallel passieren). Gewünschte Zusätze (Zucker- und Eiswürfelbereiter) können jederzeit parallel dem Getränk hinzugegeben werden (Voraussetzung: Der Automat hat bereits einen Becher ausgelassen). Die Notation "Wasserkocher + (Kaffeemühle → Aufgusschub ($P = 1$))" bedeutet, dass der Wasserkocher parallel zur Kaffeemühle und dem Aufgusschub verwendet werden darf - diese Komponenten sind unabhängig voneinander.

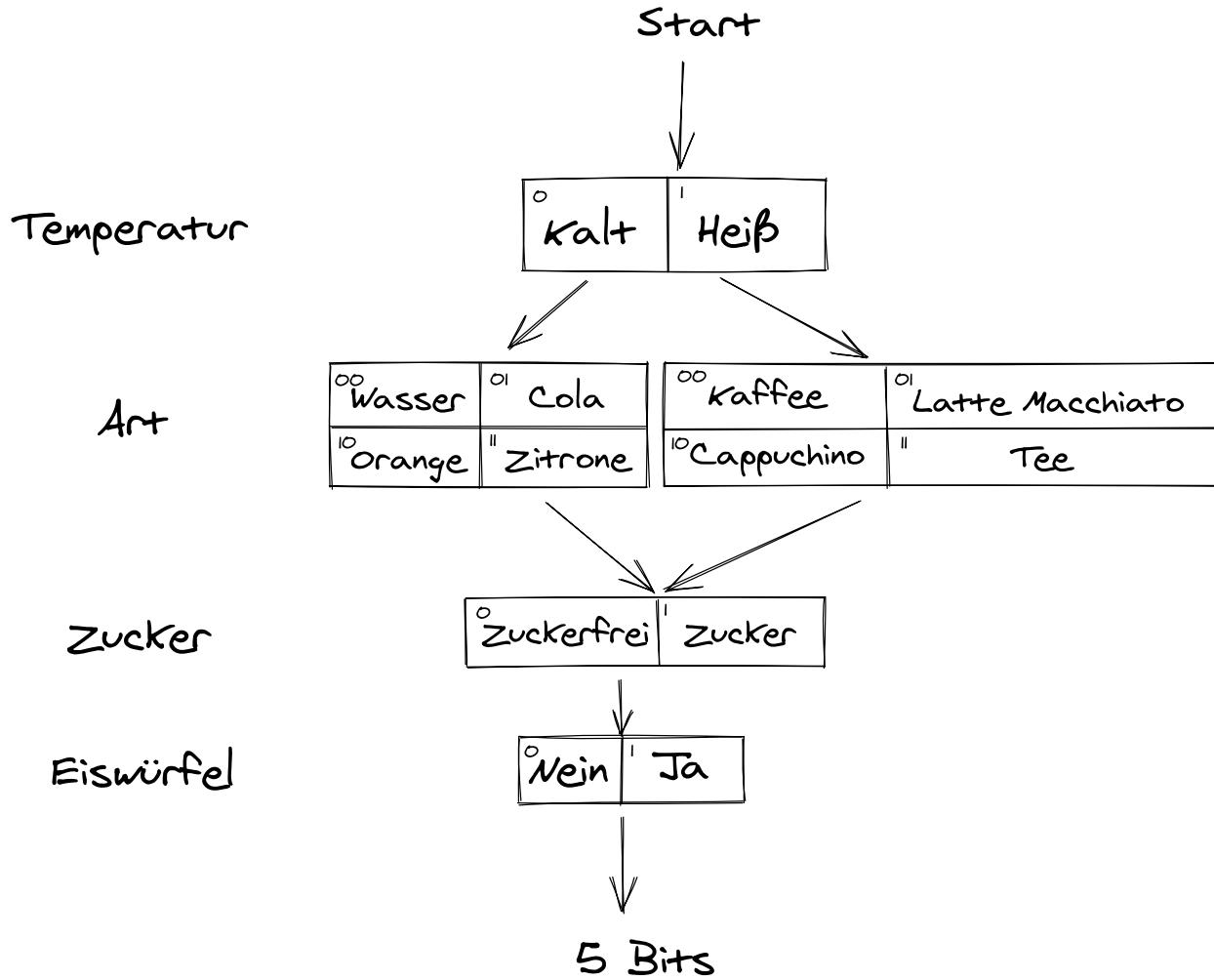
2.2 Getränkeautomat (10 PP)

In dieser Aufgabe sollen Sie den Getränkeautomaten planen und Teile davon konstruieren.

a)

Der Getränkeautomat bietet 0,5 l Getränke in verschiedenen Varianten an, deren Zubereitungen in Tabelle 1 beschrieben sind. Neben Wasser gibt es die Geschmacksrichtungen Cola, Orange und Zitrone, die zu 20 % aus Sirup besteht. Die drei Geschmacksrichtungen gibt es auch als zuckerlose Alternative. Auf Wunsch kann der Kunde diese kalten Getränke mit Eiswürfeln bestellen. Neben kalten Getränken bietet der Automat auch Kaffee, Latte Macchiato, Cappuccino und Tee an. Latte Macchiato und Cappuccino enthalten von der Rezeptur her 50 % aufgeschäumte Milch. Zur Vereinfachung nehmen Sie an, dass zunächst der Kaffee und im Anschluss die aufgeschäumte Milch ausgelassen wird. Auf Wunsch kann ein heißes Getränk mit Zucker angereichert werden. Definieren Sie eine Kodierung, die alle Getränkekodierungen abdeckt. Existieren in Ihrer Lösung Codes, die ein ungültiges Getränk kodieren. Wenn ja, nennen Sie diese und beschreiben Sie, wofür sie alternativ sinnvoll verwendet werden könnten. (2 PP)

Für eine Kodierung können wir uns eine Art Baum vorstellen:



Wir kodieren also alle Getränke in 5 Bits.

Ein paar Beispiele:

Wasser mit Eiswürfel erhält also bspw den Code 00001, aufgeschlüsselt:

0 - Kalt

00 - Wasser

0 - Zuckerfrei

1 - mit Eiswürfel

Cappuchino mit Zucker: 11010

Zuckerfreie Zitronenlimo ohne Eis: 01100

(In der nächsten Aufgabe gibt es eine vollständige Auflistung aller möglichen Kombinationen, allerdings ohne den Klartext für was sie stehen)

Natürlich gibt es ein paar Schwachsinnige Kodierungen nämlich alle heißen Getränke mit Eiswürfeln und Wasser mit Zucker, also 100X1;101X1;110X1;111X1;0001X. Diese könnten verwendet werden um bspw. Statusmeldungen an der Maschine anzeigen zu lassen wie den Füllstand der Tanks.

2 p.

b)

In dieser Aufgabe sollen Sie eine zweistufige Pipeline mit maximal möglicher Taktfrequenz entwerfen. Die Pipeline nimmt als Input die Getränkekodierung aus Teilaufgabe a) an und gibt ein einzelnes Signal mit Wert 1 aus, wenn das Getränk fertiggestellt ist. Achten Sie darauf, dass das Wasser für heiße Getränke aufgeheizt wird und das Wasser für kalte Getränke abgekühlt ist. Für den Entwurf der Pipeline stehen Ihnen alle in Tabelle 2 gelisteten Komponenten zur Verfügung.

Verwenden Sie die Komponenten so sparsam wie möglich. Die R-Ausgänge der Komponenten werden nach jedem Takt wieder auf 0 zurückgesetzt. Sobald der S t Eingang einer Komponente gesetzt wurde, müssen Sie nicht dafür sorgen, dass der S t Eingang nach Fertigstellung wieder auf eine 0 zurückgesetzt wird. Gehen Sie zur Vereinfachung außerdem davon aus, dass die Wasser-/Sirup-/Milchtänke jederzeit gefüllt sind und Becherstau oder sonstigen Störungen in Ihrer Lösung nicht abgefangen werden müssen. (8 PP)

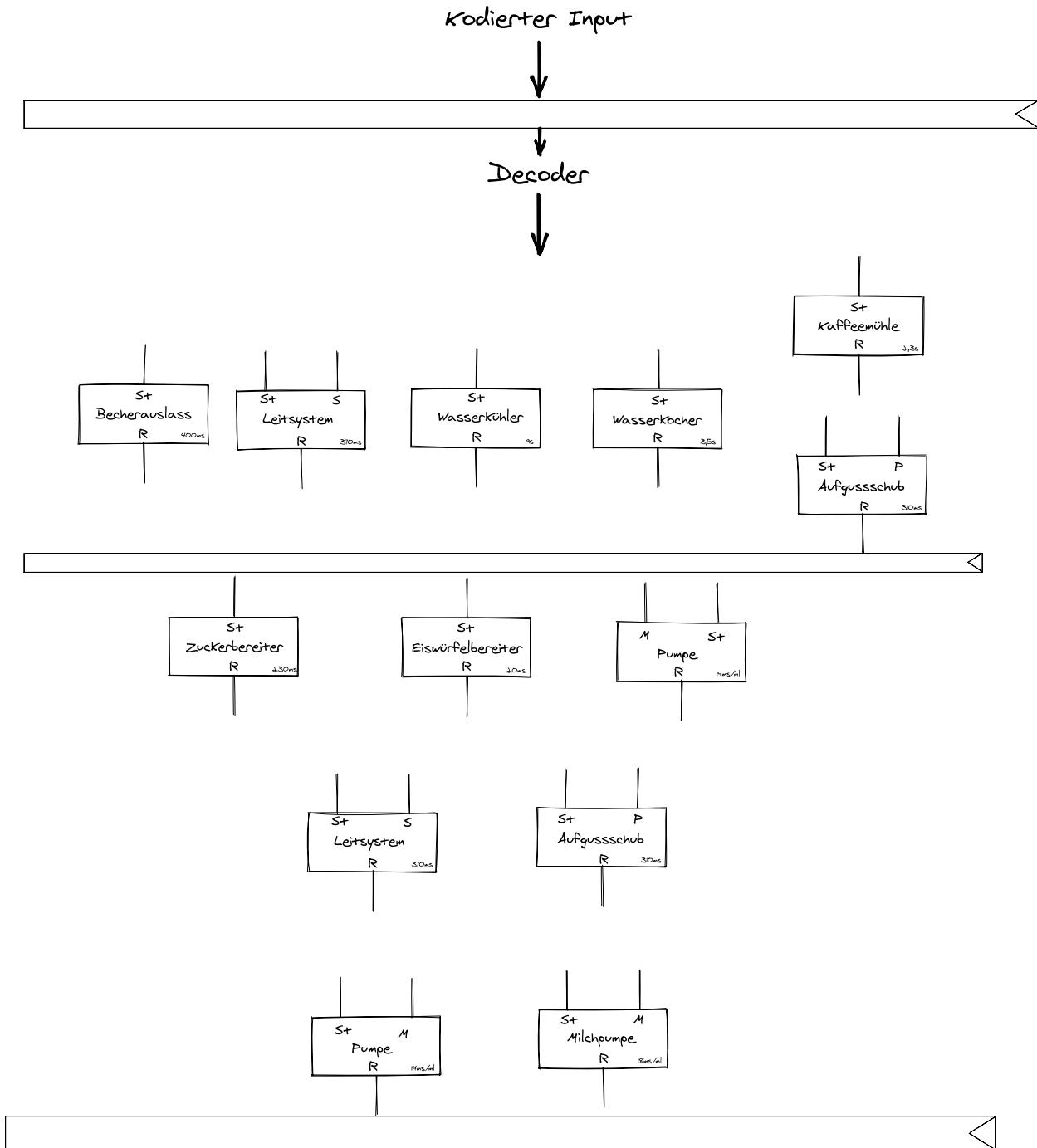
Hinweis: Überlegen Sie zunächst, welche Getränke die längste Zubereitungszeit haben und leiten Sie daraus ab, welche Komponenten Sie in welchen Pipelinestufen verwenden. Es kann sinnvoll sein, einen Dekodierer (ähnlich zu Projektaufgabe 1.3) zu entwerfen (inklusive Schaltung), der am Anfang der ersten Pipelinestufen die Getränkekodierung nimmt und daraus verschiedene Flags berechnet, die die Kontrolle der Getränkezubereitung vereinfachen (z.B. könnte man einen Ausgang des Dekodierers auf 1 setzen, sofern ein heißes Getränk zubereitet wird). Für eine korrekte Lösung benötigen Sie neben den in Tabelle 2 gelisteten Komponenten in jedem Fall kombinatorische Logik, um nur die tatsächlich benötigten Komponenten anzusprechen, sowie um auf die Fertigstellung vorheriger Komponenten zu warten (jeweils gesetzter R Ausgang der Komponente).

Name	tpd	Eingänge	Ausgänge	Beschreibung
Leitsystem	370 ms	St, S	R	Das Leitsystem lenkt die Flüssigkeitszufuhr vom Quelleingang $S \in \{00, 01, 10, 11\}$ zum Auslass wenn am Eingang St eine logische 1 anliegt. Als Quelle kann der Wassertank sowie Cola-, Orangen- und Zitronensirup gewählt werden. Wenn der Prozess abgeschlossen ist, liegt am Ausgang R eine logische 1 an.
Pumpe	14 ms/ml	St, M	R	Die Pumpe pumpt die via Eingang M in ml angegebene Menge Flüssigkeit durch das Leitsystem, sobald am Eingang St eine logische 1 anliegt. Der Ausgang R wird auf eine logische 1 gesetzt, sobald der Pumpvorgang abgeschlossen ist.
Eiswürfelbereiter	120 ms	St	R	Der Eiswürfelbereiter lässt Eiswürfel in den Becher, sobald am Eingang St eine logische 1 anliegt. Wenn der Vorgang abgeschlossen ist, liegt am Ausgang R eine logische 1 an.

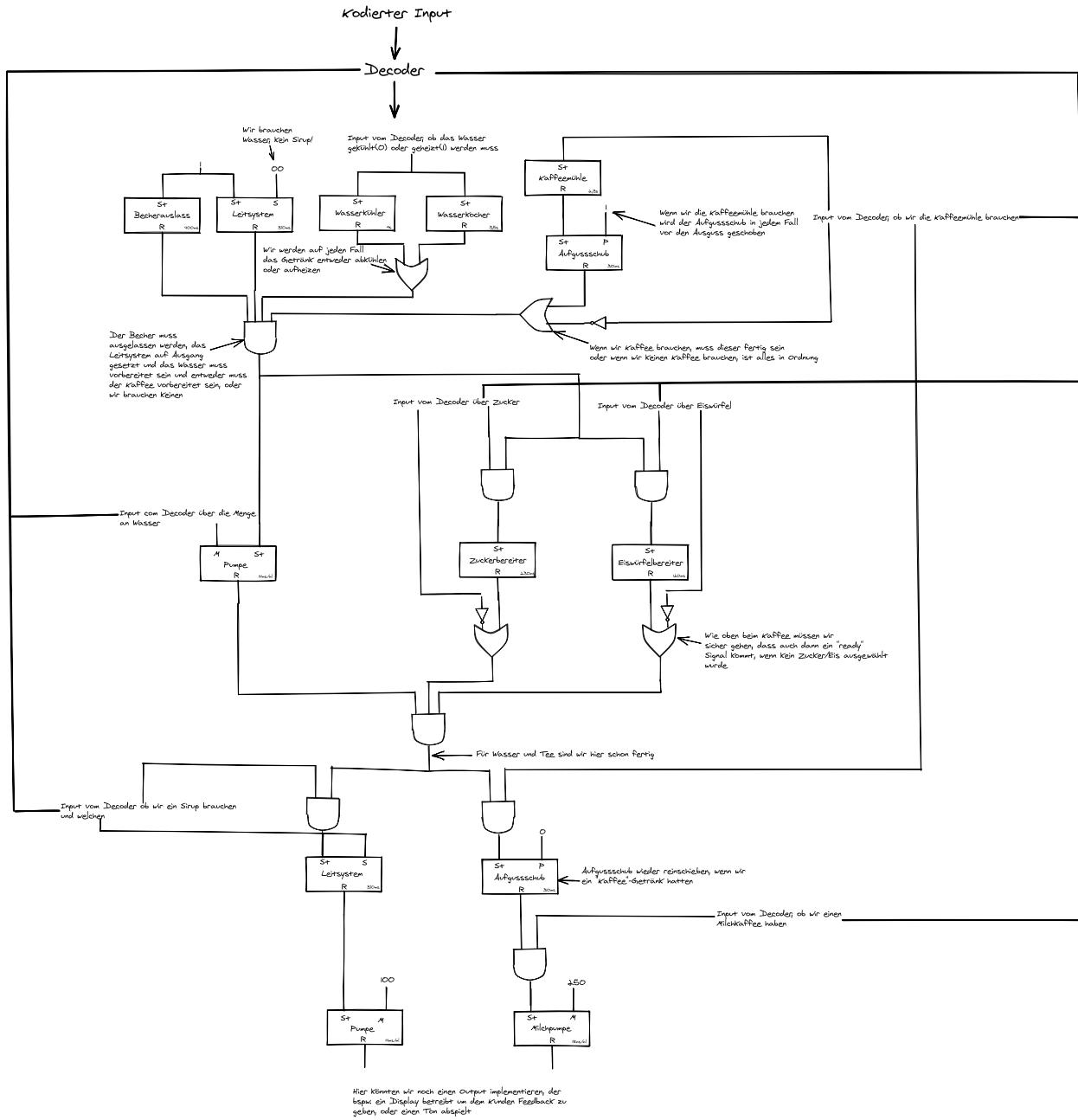
Name	tpd	Eingänge	Ausgänge	Beschreibung
Becherauslass	400 ms	St	R	Der Becherauslass lässt einen Becher aus, wenn am Eingang St eine logische 1 anliegt. Am Ausgang R liegt eine logische 1 an, sobald der Vorgang abgeschlossen ist.
Wasserkocher	3,5 s	St	R	Der Wasserkocher kocht das Wasser im Wassertank, wenn am Eingang St eine logische 1 anliegt. Sobald der Vorgang abgeschlossen ist, liegt am Ausgang R eine logische 1 an.
Wasserkühler	9 s	St	R	Der Wasserkühler kühlst das Wasser auf 7 °C, wenn am Eingang St eine 1 anliegt und gibt am Ausgang R eine 1 aus, wenn er damit fertig ist.
Milchpumpe	18 ms/ml	St, M	R	Die Milchpumpe pumpt die am Eingang M in ml angegebene Menge aufgeschäumte Milch vom Milchbehälter zum Auslass, sobald am Eingang St eine logische 1 anliegt. Am Ausgang R wird eine logische 1 ausgegeben, wenn der Vorgang abgeschlossen ist.
Zuckerbereiter	230 ms	St	R	Der Zuckerbereiter lässt eine Portion Zucker zum Auslass, wenn am Eingang St eine logische 1 anliegt. Sobald der Vorgang abgeschlossen ist, liegt am Ausgang R eine logische 1 an.
Kaffeemühle	2,3 s	St	R	Die Kaffeemühle mahlt eine Portion Kaffeebohnen in den Kaffeeaufguss, sobald am Eingang St eine logische 1 anliegt. Sobald der Vorgang abgeschlossen ist, liegt am Ausgang R eine logische 1 an.
Aufgusssschub	310 ms	St, P	R	Der Aufgusssschub bewegt den Kaffeeaufguss zwischen Kaffeemühle ($P = 0$) und dem Flüssigkeitsauslass ($P = 1$), wenn am Eingang St eine logische 1 anliegt. Sobald der Vorgang abgeschlossen ist, liegt am Ausgang R eine logische 1 an

Tabelle 2: Liste der einzelnen Komponenten des Getränkeautomaten.

Zuerst können wir uns der Übersichtshalber alle Bauteile in der richtigen Ebene darstellen um eine Parallelität zu gewährleisten.



Danach können wir erste Logik implementieren:



Jetzt brauchen wir nur noch die Logik des Decoders aufschreiben und wir sind fertig.

Folgende Werte muss der Decoder erzeugen:

- Kalt(0) / Heißgetränk(1) [Tmp]
- Kein Kaffeetrink(0) / Kaffeetrink(1) [Kf]
- Wassermenge für das erste mal pumpen - Um das in die Tabelle aufnehmen zu können teilen wir das auf in:
 - Wassermenge500 [500]
 - Wassermenge400[400]
 - Wassermenge250[250]
- Kein Zucker(0) / Zucker(1) [Zkr]
- Kein Eis(0) / Eis(1) [Eis]
- Kein Sirup(0) / Sirup(1) [Srp]
- Welchen Sirup (Cola/Orange/Zitrone) - Auch hier müssen wir aufteilen:
 - Sirupsorte_Bit1 [Bt1]
 - Sirupsorte_Bit0[Bt2]
- Kein Milchkaffee(0) / Milchkaffee(1) [MKf]

Getränkecode	Tmp	Kf	500	400	250	zKR	Eis	Srp	Bt1	Bt2	MKf
00000	0	0	1	0	0	0	0	0	-	-	0
00001	0	0	1	0	0	0	1	0	-	-	0
00010	0	0	1	0	0	1	0	0	-	-	0
00011	0	0	1	0	0	1	1	0	-	-	0
00100	0	0	0	1	0	0	0	1	0	1	0
00101	0	0	0	1	0	0	1	1	0	1	0
00110	0	0	0	1	0	1	0	1	0	1	0
00111	0	0	0	1	0	1	1	1	0	1	0
01000	0	0	0	1	0	0	0	1	1	0	0
01001	0	0	0	1	0	0	1	1	1	0	0
01010	0	0	0	1	0	1	0	1	1	0	0
01011	0	0	0	1	0	1	1	1	1	0	0
01100	0	0	0	1	0	0	0	1	1	1	0
01101	0	0	0	1	0	0	1	1	1	1	0
01110	0	0	0	1	0	1	0	1	1	1	0
01111	0	0	0	1	0	1	1	1	1	1	0
10000	1	1	1	0	0	0	0	0	-	-	0
10001	1	1	1	0	0	0	1	0	-	-	0

Getränkecode	Tmp	Kf	500	400	250	zKR	Eis	Srp	Bt1	Bt2	MKf
10010	1	1	1	0	0	1	0	0	-	-	0
10011	1	1	1	0	0	1	1	0	-	-	0
10100	1	1	0	0	1	0	0	0	-	-	1
10101	1	1	0	0	1	0	1	0	-	-	1
10110	1	1	0	0	1	1	0	0	-	-	1
10111	1	1	0	0	1	1	1	0	-	-	1
11000	1	1	0	0	1	0	0	0	-	-	1
11001	1	1	0	0	1	0	1	0	-	-	1
11010	1	1	0	0	1	1	0	0	-	-	1
11011	1	1	0	0	1	1	1	0	-	-	1
11100	1	0	1	0	0	0	0	0	-	-	0
11101	1	0	1	0	0	0	1	0	-	-	0
11110	1	0	1	0	0	1	0	0	-	-	0
11111	1	0	1	0	0	1	1	0	-	-	0

Die 5 Bits können wir in $s_4 - s_0$ zerlegen und erhalten damit folgendes (Da wir die Codierung schlau gewählt haben, fällt uns das Decodieren einfach):

Temperatur = s_4

Kaffeetrinken = $s_4 \bar{s}_3 \bar{s}_2$

Wassermenge500 = $\bar{s}_4 \bar{s}_3 \bar{s}_2 + s_4 \bar{s}_3 \bar{s}_2 + s_4 s_3 s_2$

Wassermenge400 = $\bar{s}_4 s_2 + \bar{s}_4 s_3$

Wassermenge250 = $s_4(s_3 \oplus s_2)$

Zucker = s_1

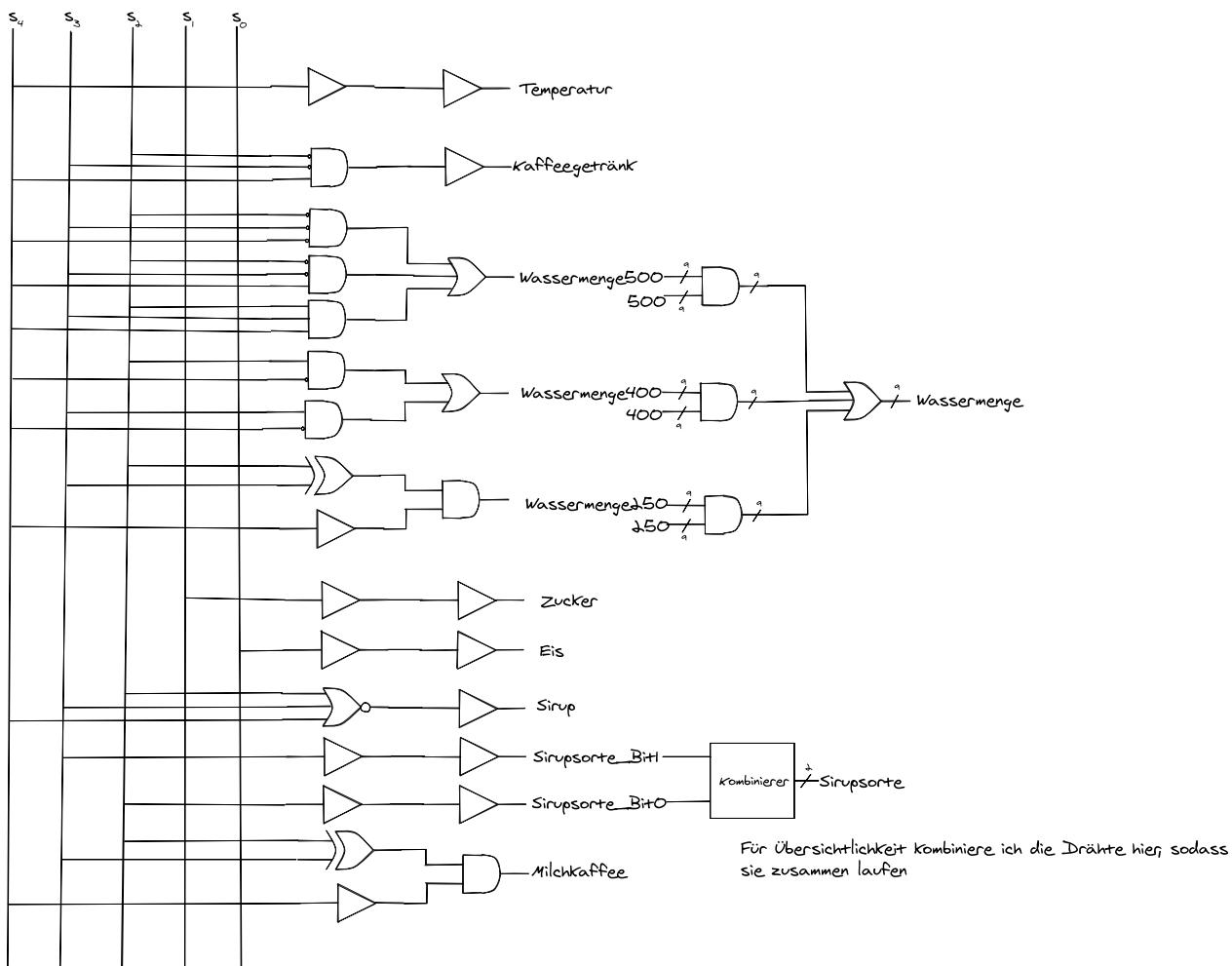
Eis = s_0

Sirup = $\bar{s}_4 \bar{s}_3 \bar{s}_2 = \bar{s}_4 + s_3 + s_2$

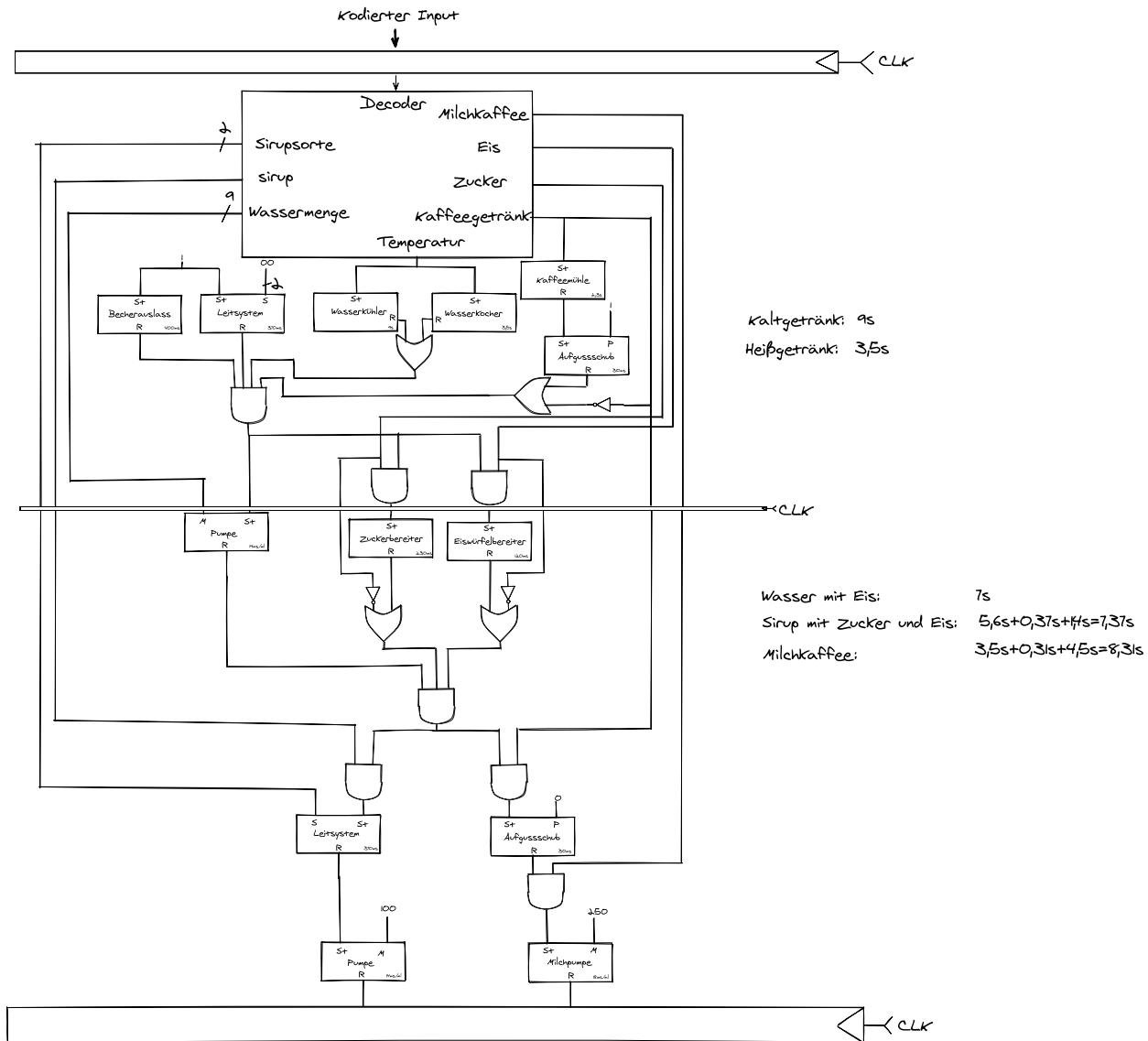
Sirupsorte_Bit1 = s_3

Sirupsorte_Bit0 = s_2

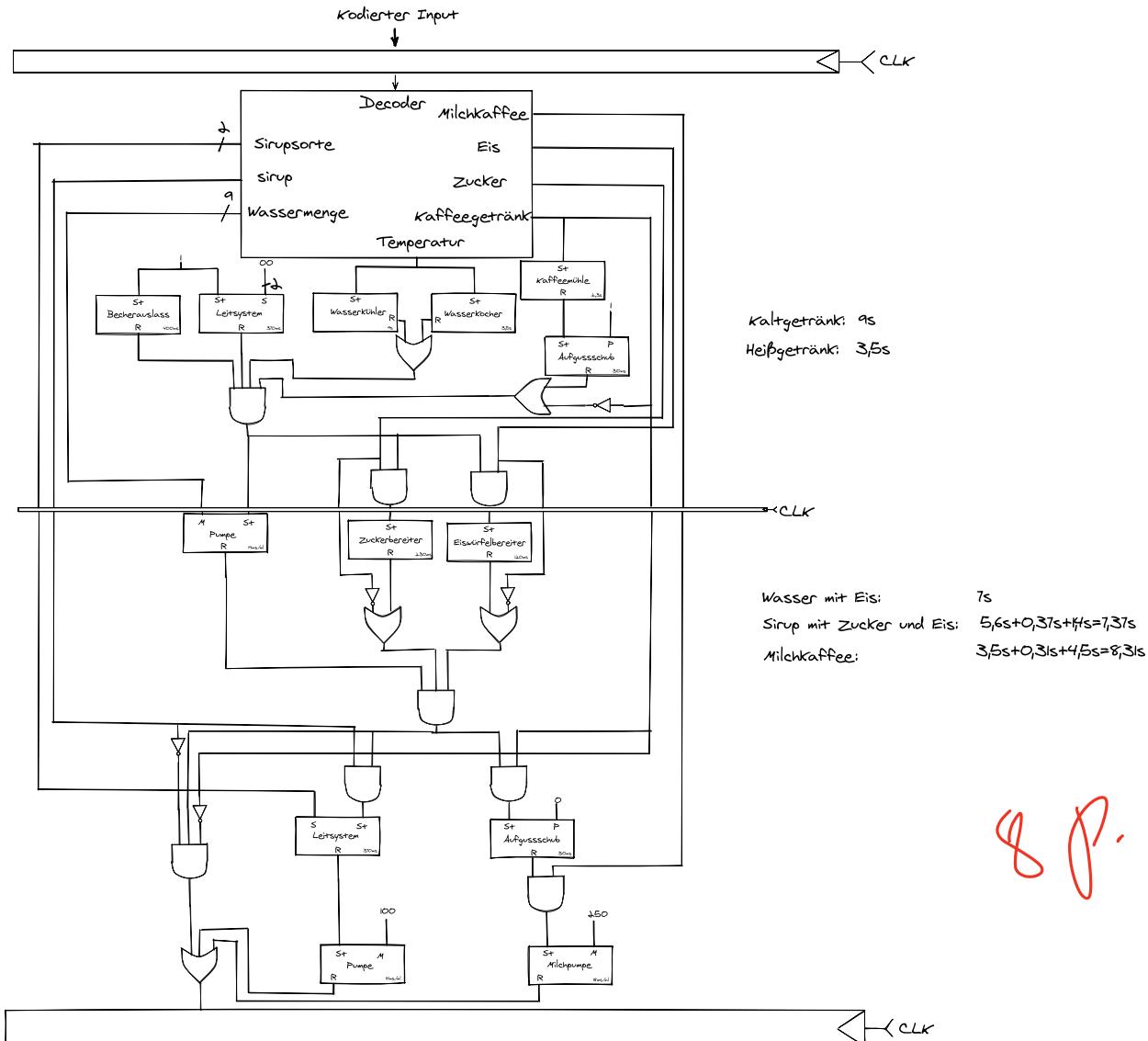
Milchkaffee = $s_4(s_3 \oplus s_2) = \text{Wassermenge250}$



Jetzt müssen wir nur alles zusammen setzen und die Register für die Pipelines hinzufügen:



Als letztes müssen wir noch die Outputs verbinden.



Wenn wir die Verzögerung der Gatter unbeachtet lassen (da diese um ein Vielfaches kleiner sind als die hier verwendeten Bausteine) können wir f_{CLK} ausrechnen mit

$f_{CLK} \leq \min\left(\frac{1}{9}s, \frac{1}{8,31}s\right) = 0,111Hz$, also haben wir eine Maximale Frequenz von $0,111Hz$ und eine Latenz von $2 * 9s = 18s$, also können wir 1 Getränk alle 9 Sekunden erzeugen und es dauert 18 Sekunden bis ein angefordertes Getränk von der Maschine bearbeitet wurde.

Zentrale Kontrolleinheit (9 PP)

In dieser Aufgabe entwerfen Sie die zentrale Kontrolleinheit für das System. Diese ist am Bus mittels der Eingänge X_0, X_1 und Ausgänge Y_0, \dots, Y_{n-1} angeschlossen. Von der zentralen Kontrolleinheit erhält der Bus für jede Komponente, an die er angeschlossen ist, bis zu zwei weitere Steuersignale, die nicht auf dem Schaltplan im Prolog eingezeichnet sind und angeben, ob die jeweilige Komponente senden oder empfangen soll (oder nichts von beidem, wenn beide Signale aus sind). Benennen Sie diese Ausgänge C_{in}, C_{out} für die Kontrolleinheit, A_{in} für das Display, S_{in}, S_{out} für die Schiebetür, L_{in}, L_{out} für die Lichtschranke, und G_{in} für den Getränkeautomaten. Hierbei wird das in-Signal der

jeweiligen Komponente auf eine logische 1 gesetzt, wenn die Komponente Daten vom Bus empfangen soll und das out-Signal, wenn die Komponente Daten senden soll.

Der Schalter, bei dem die Kunden ihr Ticket scannen, ist direkt an die Kontrolleinheit angeschlossen. Dieser erhält als Eingang ein Signal R, das angibt, wenn der nächste Kunde sein Ticket scannen darf. Ein Ausgang V gibt an, wenn ein Ticket vom Kunden erfolgreich gescannt wurde und die Seriennummer im System hinterlegt ist. Darüber hinaus wird ebenfalls die Kodierung G_0, \dots, G_{n-1} vom Schalter ausgegeben, welche das gewünschte Getränks angibt, das an den Getränkeautomaten weitergegeben werden soll.

Der Ablauf des Kontrollsystens ist wie folgt:

1. Initialer Zustand: Es werden keine Daten über den Bus gesendet und die maximale Teilnehmeranzahl wird auf 50 Personen gesetzt. Diese Zahl wird bereits auf dem Display angezeigt.
2. Sofern noch Besucher passieren dürfen, wartet das Kontrollsystem auf das V -Signal des Schalters, welches ein gültiges Scannen symbolisiert. Am Eingang G ist der Getränkewunsch kodiert.
3. Der Getränkewunsch wird an den Getränkeautomaten weitergereicht.
4. Die Schiebetür öffnet sich.
5. Sobald die Schiebetür komplett geöffnet wurde, wartet das Kontrollsystem auf eine Lichtschrankenunterbrechung. Falls sich die Bewegungsrichtung der Schiebetür unerwartet ändert, liegt ein Systemfehler vor und das System fährt herunter.
6. Sobald die Lichtschranke unterbrochen wurde, wird diese zurückgesetzt und die Tür schließt sich. Auch hier gilt wieder, dass das System herunterfährt, wenn sich die Bewegungsrichtung der Schiebetür unerwartet ändert.
7. Sobald die Tür geschlossen wurde, wird die Anzeige auf dem Display aktualisiert und der nächste Besucher darf passieren. Übergeben Sie die anzuzeigende Zahl an die Display Komponente.

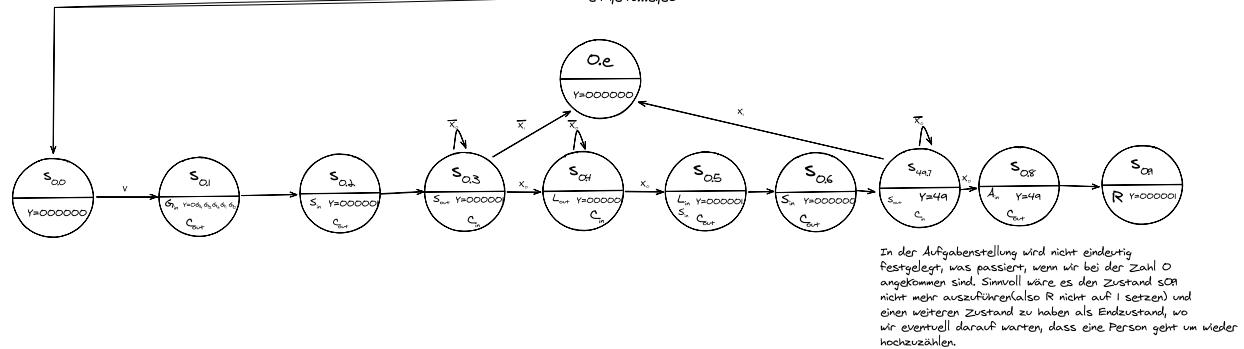
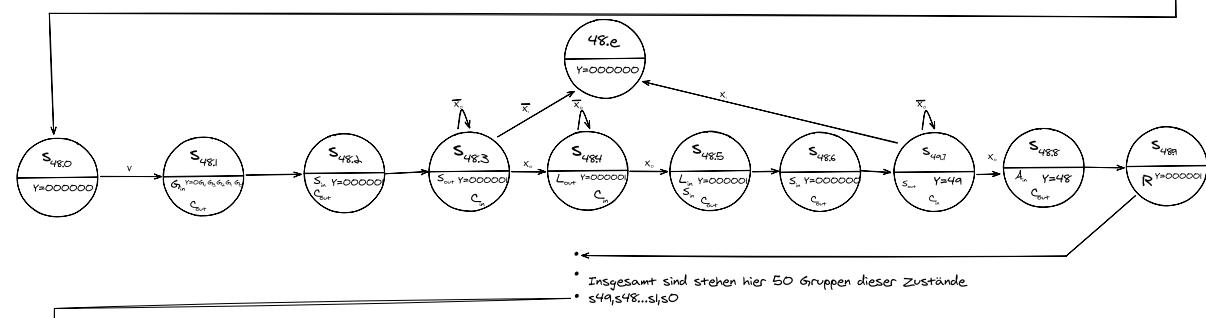
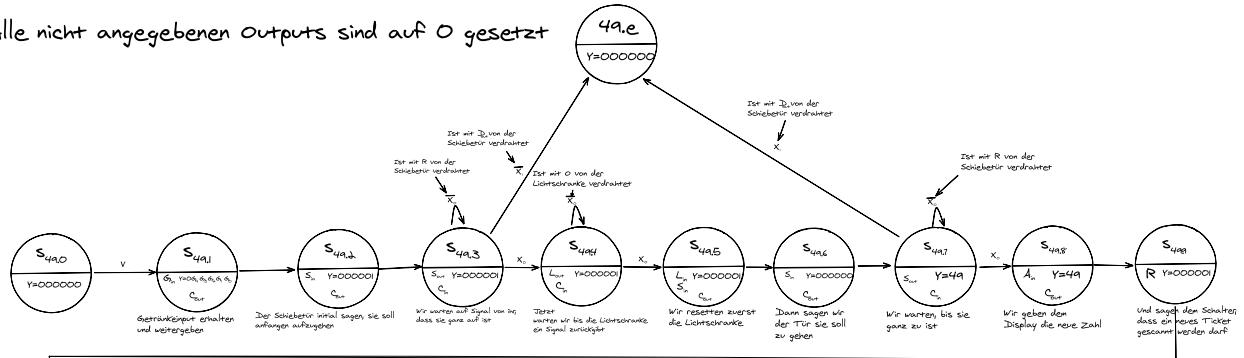
Sie müssen für alle Komponenten die korrekten Eingänge/Ausgänge am Bus setzen, wie sie in den vorherigen Aufgabenstellungen erklärt wurden.

Aufgabe:

Entwerfen Sie das **FSM-Diagramm eines Moore-Automaten**, der das zentrale Kontrollsystem wie oben beschrieben umsetzt. Die Ausgänge des Automaten sind die Steuersignale zum Bus (hier genügt es, nur die Ausgänge aufzulisten, die auf 1 gesetzt sind) sowie die Nachricht Y , die in den Bus geschickt wird. Stellen Sie den Zustand "System fährt herunter" mit einem Fehlerzustand dar, der alle Ausgänge auf 0 setzt und keine Folgezustände besitzt. Sie dürfen eine Kette von Zuständen mit Punkten abkürzen, wenn alle Zustände in der Kette dieselben Zustandsübergänge und ähnliche Ausgänge haben (vgl. Abbildung 2). Die Zustände, die den Anfang und das Ende der Kette bilden, müssen in jedem Fall gezeichnet werden. Geben Sie auch die Anzahl der Zustände an, die sie abkürzen. Weiterhin dürfen Sie alle Methoden zur Umsetzung von Zustandsautomaten verwenden, die Sie in der Vorlesung gelernt haben.

Leider habe ich erst nachdem ich schon fertig war mit allen Aufgaben außer dieser hier, dass D_{out} dauerhaft die Richtung anzeigen soll, in der sich die Tür bewegt. Ich habe es in den Moore-Automaten eingebaut, damit die Bedingung "dass das System herunterfährt, wenn sich die Bewegungsrichtung der Schiebetür unerwartet ändert" erfüllt ist. Bei 2.1 habe ich dies nicht eingebaut, da es nicht in der Aufgabenstellung gefordert war. Dort könnte man allerdings einfach D_{in} direkt mit D_{out} verdrahten und erhält so die Richtung in der die Tür sich bewegt und die finale Position in der sie sich befindet. In einer realen Implementierung wäre dies wahrscheinlich mit einem Sensor in den Motoren verwirklicht, der die "reale" Bewegungsrichtung in Form von Feedback zurückliefert und wenn irgendetwas komisch ist (entweder die reale Bewegungsrichtung anders die gewünschte oder auch die Bewegungsrichtung der beiden Motoren unterschiedlich) würde das System herunterfahren.

Alle nicht angegebenen Outputs sind auf 0 gesetzt



Bus (3 PP)

In dieser Aufgabe sollen Sie einen digitalen Schaltplan für den Bus erstellen, der das Display, die Schiebetür, den Getränkeautomaten, die Lichtschranke und die zentrale Kontrolleinheit verbindet.

a)

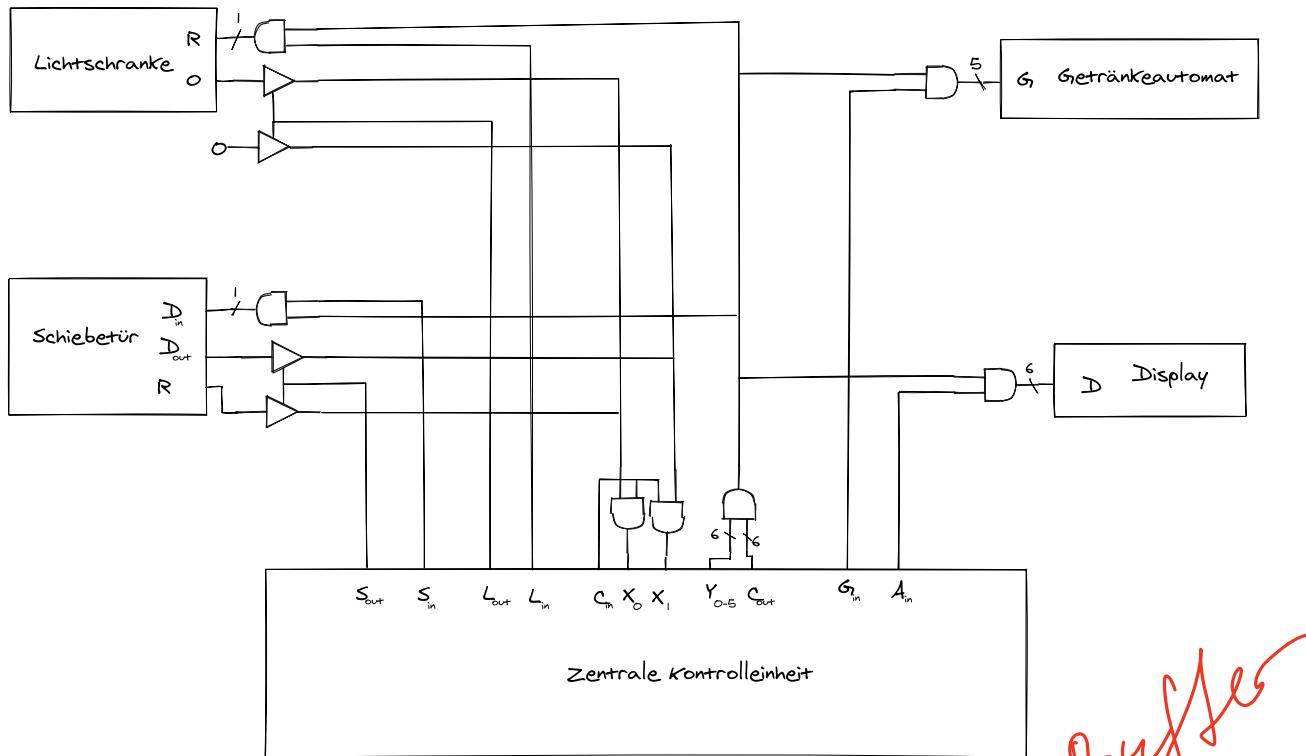
Entwerfen Sie einen digitalen Schaltplan für den Bus. Orientieren Sie sich an den Interfaces aus dem Prolog. Achten Sie darauf, dass keine Komponente unnötige Wires erhält (z.B. benötigt die

Lichtschranke weniger

Daten vom Bus als der Getränkeautomat). Beachten Sie außerdem die zusätzlichen Eingänge, die in Projekt 2.3 beschrieben sind und die bestimmen, welche Komponente empfängt und sendet. (2PP)

Wenn ein Draht von 6 auf weniger Bits an den Eingängen reduziert wird, werden immer die most important bits weggenommen, also angefangen mit Y_5

Wenn mehrere Drähte aus und-Gattern herausgehen, sind es auch immer so viele und-Gatter verwendet, damit jeder Y-Draht mit dem "in"-Draht verglichen wird.



Tristate - Buffer

1P.

b)

Ihr Schaltplan benötigt jeweils bis zu 2 Signale, um zu steuern, welche Komponente sendet und empfängt. Beschreiben Sie, wie man die Anzahl dieser Kontroll-Bits reduzieren kann. (1PP)

Wir haben insgesamt 3 "out"- und 5 "in"- Bits, die steuern welche Komponenten senden bzw. empfangen.

C_{in} können wir weg lassen, da alle Komponenten ihre Daten nur der Zentralen Kontrolleinheit senden und wir deshalb wissen, sobald ein "out"-Bit an ist, soll die Zentrale Kontrolleinheit empfangen (also können wir entweder die Zentrale Kontrolleinheit dauerhaft empfangen lassen, oder immer dann wenn irgendeiner der anderen "out"-Bits sendet, also C_{in} abhängig von L_{out} und S_{out} machen, indem wir die Output-Kontroll-Bits mit einem Oder-Gatter verbinden und dies als C_{in} verwenden).

C_{out} können wir weg lassen, da wir über die "in"-Bits schon steuern wer was empfängt, wir müssen nicht auch noch Steuern, dass über die Y-Drähte nichts gesendet wird, dort kann dauerhaft "sinnlose" Daten gesendet werden, solange keiner der anderen Komponenten diese empfängt. damit haben wir noch S_{out} und L_{out} , wo wir keine weiteren bits einsparen können, auch wenn wir beide über eine gemeinsame Verbindung laufen lassen brauchen wir immer noch 3 Zustände: Niemand sendet(00), L sendet(01), S sendet(10). Wir haben allerdings 11 frei um eine weitere, sendende Komponente hinzuzufügen.

Bei den "in"-Bits haben wir noch D_{in} , G_{in} , L_{in} , S_{in} übrig. Um hier Bits, bzw Drähte einzusparen können wir ein 3 Bit breites, gemeinsames Signal verwenden:

Signal	Effekt
000	Niemand empfängt
001	Display empfängt
010	Getränkeautomat empfängt
011	Schiebetür empfängt
100	Lichtschranke empfängt
101	Lichtschranke und Schiebetür empfangen
110	Frei
111	Frei

1 P.

Wir haben also zwei Kontroll-Bits für den Output aller Komponenten und 3 Kontroll-Bits für den Input aller Komponenten, insgesamt also 5 Kontroll-Bits, anstatt 8.

Ich hoffe Sie hatten eine besinnliche und erfrischende Weihnachtszeit!