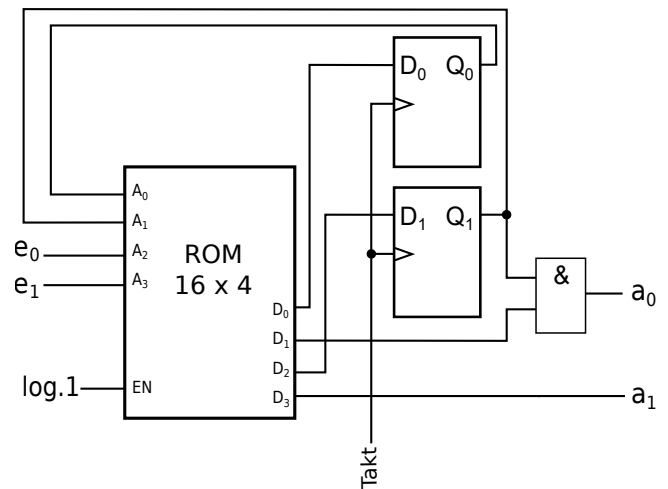


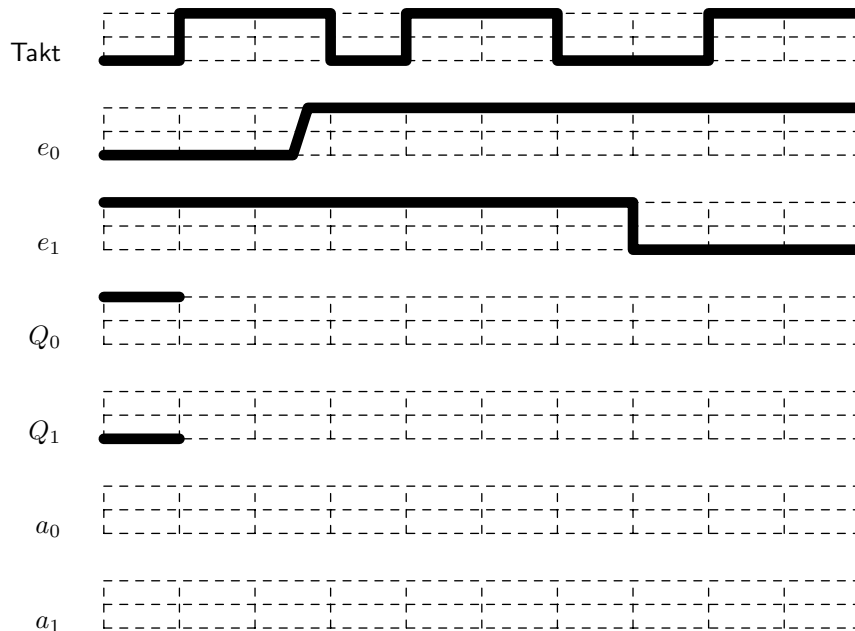
Aufgabe 1: Timing Diagramm

Gegeben ist das dargestellte Schaltwerk mit zwei Eingabesignalen e_0 und e_1 und zwei Ausgabesignalen a_0 und a_1 . Im 16×4 ROM-Baustein, der Teil der Schaltung ist, sind die in der Tabelle angegebenen Werte gespeichert:

A_3	A_2	A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1	1	0
0	0	0	1	1	0	0	1
0	0	1	0	1	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1
0	1	1	1	0	0	1	0
1	0	0	0	0	0	1	1
1	0	0	1	0	1	1	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	1	1
1	1	0	0	0	1	1	0
1	1	0	1	0	1	1	0
1	1	1	0	0	0	1	1
1	1	1	1	1	1	0	0

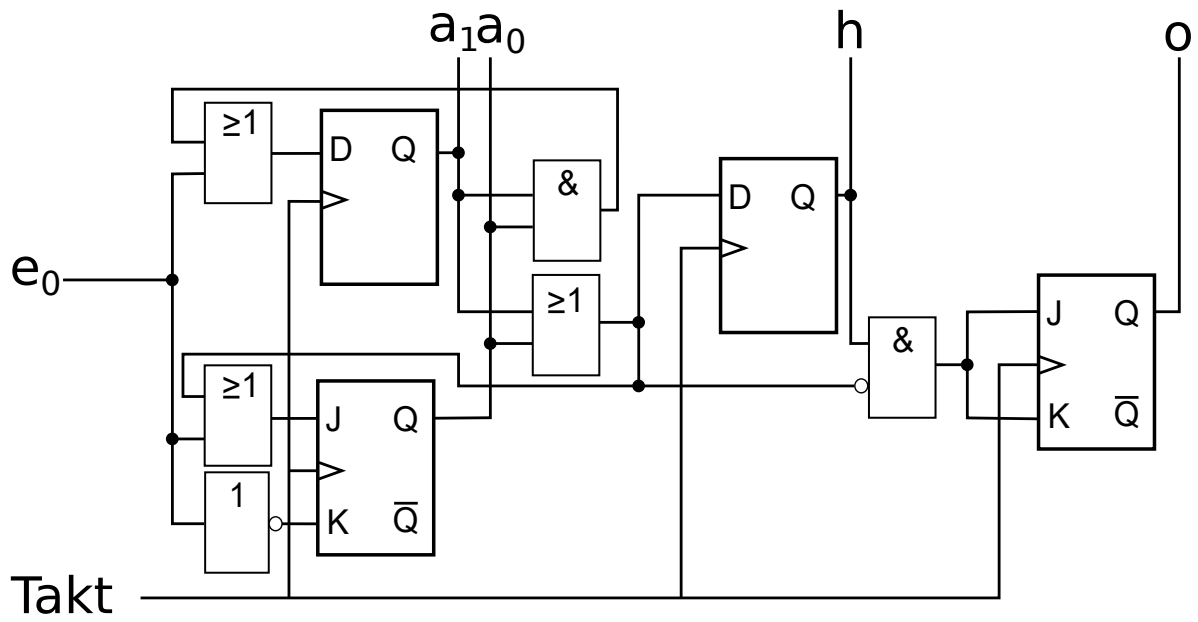


Zeichnen Sie den Verlauf der Signale Q_0 , Q_1 , a_0 und a_1 . Die Durchlaufzeiten des ROM-Bausteins sowie des AND-Gatters sollen dabei vernachlässigt werden.

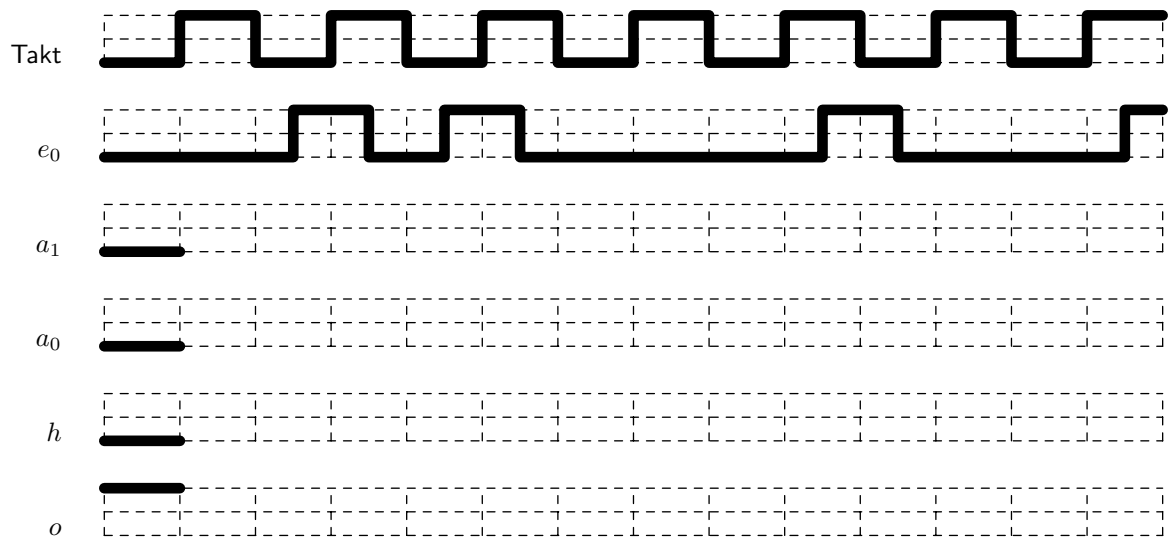


Aufgabe 2: Timing-Diagramm

Es ist folgende Schaltung gegeben:

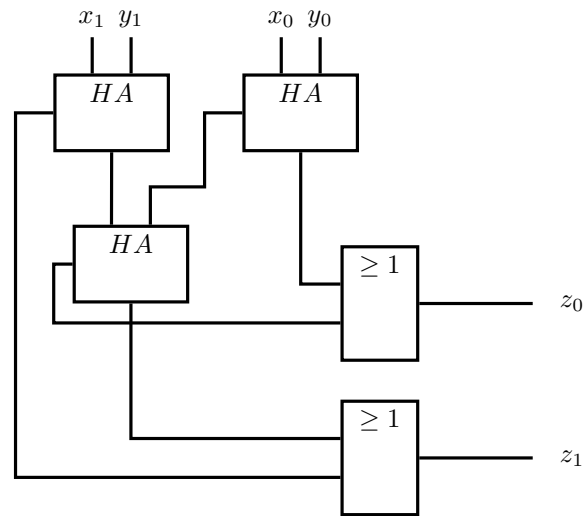


Überlegen Sie sich die Funktionsweise der Schaltung, sodass Sie diese in der Übung erklären können und vervollständigen Sie das nachfolgende Timing-Diagramm. Achten Sie dabei auf die Flankentriggerung.



Aufgabe 3: Schaltung zu ROM

Gegeben ist folgendes Schaltbild, das aus Halbaddierer-Bausteinen sowie einem Gatter besteht. Der linke Ausgang des HA ist C_{out} .

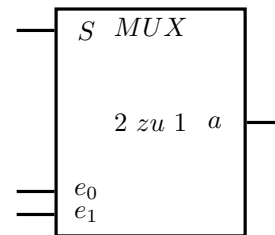
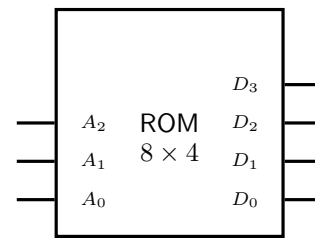


- a) Analysieren Sie die oben gegebene Schaltung und vervollständigen Sie den Vordruck der Wahrheitstabelle.

x_1	y_1	x_0	y_0	z_0	z_1
0	0	0	0		
0	0	0	1		
0	0	1	0		
0	0	1	1		
<hr/>					
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
<hr/>					
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
<hr/>					
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

b) Realisieren Sie die Wahrheitstabelle aus Aufgabe a) mit den unten gegebenen Bausteinen (Sie können den Multiplexer mehrmals verwenden, den ROM jedoch nur einmal). Geben Sie außerdem den Inhalt des ROMs an.

A_2	A_1	A_0	D_3	D_2	D_1	D_0
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

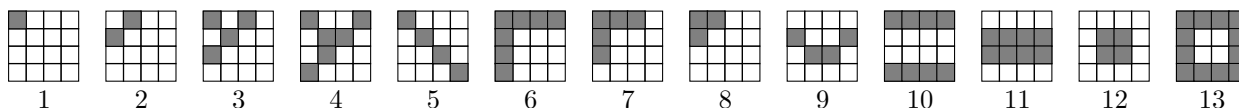


Aufgabe 4: Entwurf & Realisierung einer Boole'schen Funktion via PLA

Entwerfen Sie einen Teil eines Decoders, welcher die dargestellte 4x4 LED-Matrix ansteuert. Die LED-Matrix besitzt 16 Eingänge (s_0 bis s_{15}), welche jeweils die angegebene LED kontrollieren (wobei eine LED leuchtet wenn am entsprechenden Eingang logisch 1 anliegt).

s_0	s_1	s_2	s_3
s_4	s_5	s_6	s_7
s_8	s_9	s_{10}	s_{11}
s_{12}	s_{13}	s_{14}	s_{15}

Der Decoder besitzt die Eingänge e_3, e_2, e_1, e_0 , welche ein auf der LED-Matrix darzustellendes Muster codieren (e_3 ist das MSB). Die verfügbaren Muster sind unten angegeben. Grau eingefärbte Felder symbolisieren leuchtende LEDs, weiße Felder inaktive LEDs. Die Binärcodierung der unter dem Muster dargestellten Zahl entspricht der Codierung des Musters. Entspricht die in e_3, e_2, e_1 und e_0 dargestellte Zahl keinem Muster, soll kein LED leuchten.



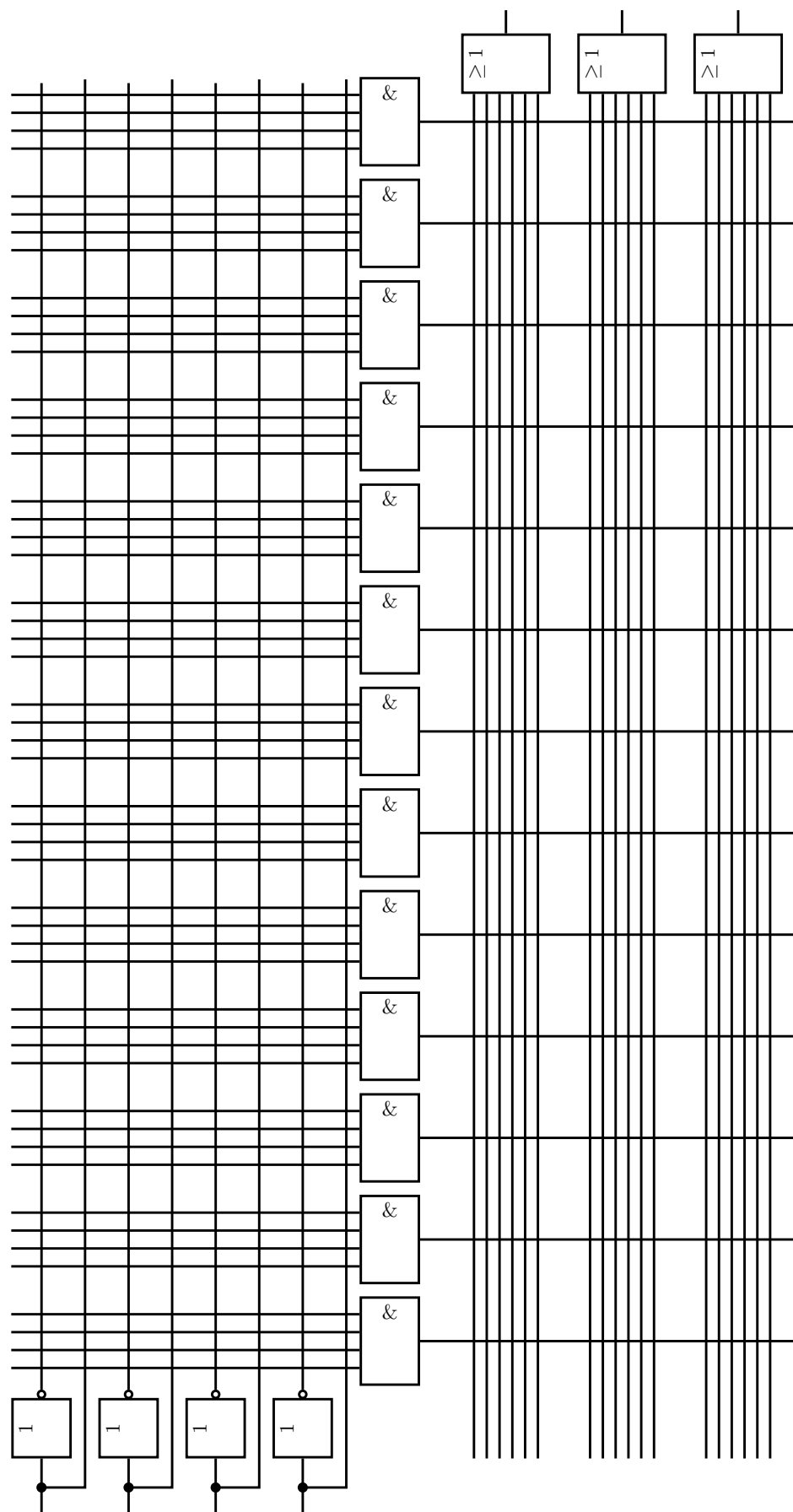
Realisieren Sie eine **minimale** Schaltung für die LEDs an den Positionen s_0, s_4, s_{10} (d.h. die drei Ausgänge Ihrer Schaltung sollen die Werte für diesen drei Eingängen der LED-Matrix entsprechen).

- a) Vervollständigen Sie die nachfolgende Wahrheitstabelle. Leiten Sie eine boolesche Funktion in der notwendigen Minimalform von der Wahrheitstabelle ab. Sie benötigen diese für Unteraufgabe b). Sie können dafür beliebige Werkzeuge verwenden. Wir empfehlen den Karnaugh-Veitch Map Rechner¹ entwickelt an der Uni Marburg. Sie müssen die Funktionsweise des verwendeten Werkzeuges nicht erklären. Nur die Eigenschaften des Ergebnisses (Normalform).

e_3	e_2	e_1	e_0	s_0	s_4	s_{10}	Ausgang „ s_0 “:
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				Ausgang „ s_4 “:
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				Ausgang „ s_{10} “:
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

¹<https://www.mathematik.uni-marburg.de/~thormae/lectures/t11/code/karnaughmap/>

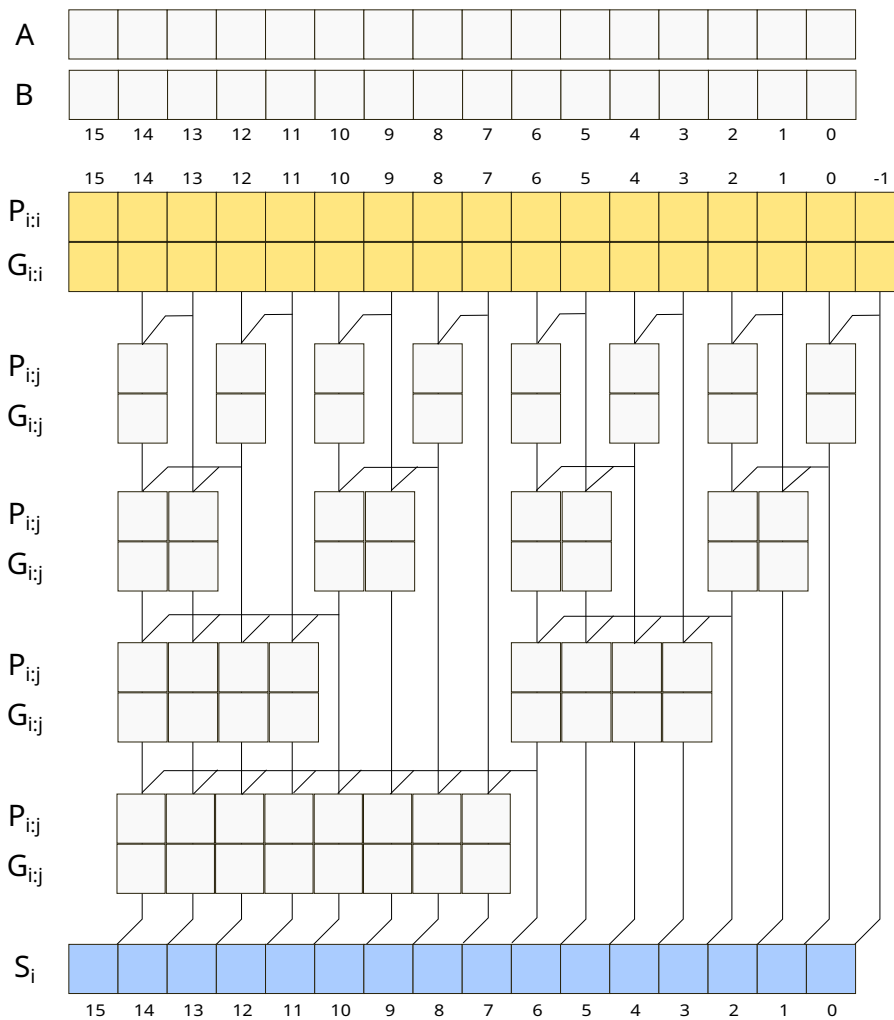
b) Realisieren Sie den Decoder für die drei Werte s_0 , s_4 und s_{10} im unten dargestellten PLA. Vergessen Sie nicht Ein- und Ausgänge zu beschriften.



Aufgabe 5: Prefix-Addierer

a) Erklären Sie, wie ein Prefix-Addierer die Addition implementiert. Gehen Sie dabei auf die G bzw. P Signale ein.

b) Gegeben ist die folgende Skizze eines Prefix-Addierers. Berechnen Sie $(15231)_{10} + (-4713)_{10}$ indem Sie die untenstehende Skizze mit den konkreten Werten befüllen. Nehmen Sie $C_{in} = 0$ an.



Aufgabe 6: Geschwindigkeit von Addierern

Wie Sie in der Vorlesung gehört haben, gibt es verschiedene Implementierungen von Addierern. In dieser Übung werden Sie sich die Geschwindigkeit (Latenz) von den vorgestellten Addierern näher ansehen.

- a) Bevor Sie die Addierer vergleichen können, sollten Sie die Formeln für die Latenz der Addierer verstehen. Unten finden Sie die in der Vorlesung präsentierten Formeln. Erklären Sie diese für jeden Addierer. Falls notwendig, zeichnen Sie sich Skizzen der Schaltungen, um die Formeln besser erklären zu können.

- Ripple-Carry-Addierer: $t_{ripple} = Nt_{FA}$

- Carry-Lookahead-Addierer: $t_{CLA} = t_{PG} + t_{PG_{Block}} + \left(\frac{N}{k} - 1\right)t_{and\&or} + kt_{FA}$

- Prefix-Addierer: $t_{PA} = t_{PG} + \log_2 N(t_{PG_{Prefix}}) + t_{xor}$
Erklären Sie auch wieso der letzte Summand nicht $2t_{xor}$ ist.

- b) Nehmen Sie an, dass ein Volladdierer eine Latenz von $300ps$ hat. Weiters, nehmen Sie an, dass Gatter mit 2 Eingängen eine Latenz von $200ps$ haben. Berechnen Sie, ab welcher Anzahl an Bits (N) ein Carry-Lookahead-Addierer ($k = 4$) schneller ist, als ein Ripple-Carry-Addierer. Betrachten Sie nur N die durch k teilbar sind.

Aufgabe 7: Funktionen einer ALU

Diese Aufgabe bezieht sich auf die ALU aus der Vorlesung (siehe `03_building_blocks.pdf`, Folie 34 ff.). Seien Sie sich bewusst, dass Sie die Lösung anhand Ihrer Abgabe erklären müssen. Machen Sie daher Skizzen zum Aufbau der ALU, falls das dazu notwendig ist.

- a) Erklären Sie wie die ALU eine Subtraktion umsetzt. Gehen Sie dabei auch auf die Notwendigkeit der Verbindung zwischen `ALUControl0` und dem Addierer ein.

- b) Erklären Sie wie die ALU die vier Status Ausgänge (V, C, N, Z) umsetzt. Legen Sie den Fokus der Erklärung auf das `overflow flag`.

Aufgabe 8: ROM Erweiterung

Die untenstehenden Aufgaben fordern Sie dazu auf, ROM Bausteine aus anderen ROM Bausteinen zu konstruieren. Die drei untenstehenden Abbildungen zeigen alle möglichen Bausteine, die Sie brauchen. Beachten Sie, dass Sie **nur jene Bausteine verwenden dürfen, die in der Aufgabe beschrieben werden**.

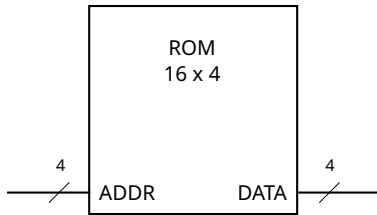


Abbildung 1: 16 x 4 ROM Baustein

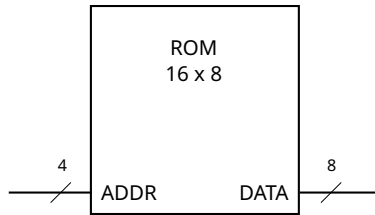


Abbildung 2: 16 x 8 ROM Baustein

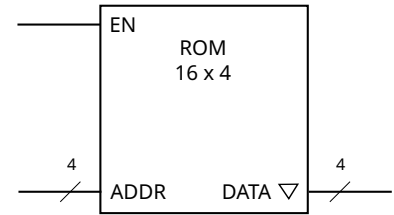


Abbildung 3: 16 x 4 ROM Baustein mit Enable-Eingang

- a) Ihnen stehen zwei 16x4 ROM Bausteine zur Verfügung. Kombinieren Sie diese zu einem 32x4 ROM Baustein in dem Sie einen Multiplexer mit 4-bit breiten Datensignalen verwenden.

b) Ihnen stehen ein Inverter und zwei 16x4 ROM Bausteine mit enable Eingang und Tri-State Ausgang zur Verfügung. Kombinieren Sie diese zu einem 32x4 ROM Baustein, ohne weitere Schaltelemente zu verwenden.

c) Ihnen steht ein 16x8 ROM zur Verfügung. Wie können Sie diesen mit Hilfe eines Multiplexers zu einem 32x4 ROM umwandeln?