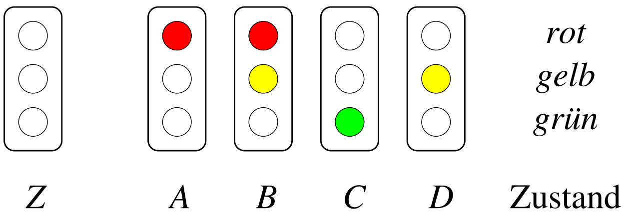
	Praxistutorium **Grundzüge der Informatik B1**	**Aufgabenblatt 4**

Bei den folgenden drei Aufgaben sollen vereinfachte Ampelschaltungen entworfen werden, wobei die verschieden Aufgaben unterschiedliche Aspekte des Entwurfs von Automaten durch praktische Realisierung verdeutlichen sollen.



Verwenden Sie zur Realisierung von Zustandsvariablen in HADES möglichst Flipflops vom Typ DFFR (D-Flipflop *mit Reset*). Da der Simulator mit einer mehrwertigen Logik operiert, wird der `NR`-Anschluß benötigt, um die Bauteile überhaupt auf einen definierten Wert zurücksetzen zu können. Die dafür vorgesehene Komponente ist der `power-on reset`-Schalter (Menü `IO`), der nach Neustart der Simulation automatisch für das Zurücksetzen der angeschlossenen Flipflops sorgt.
Generell ist der mit 0-Bits codierte Zustand also der Startzustand des jeweils implementierten Automaten, da nach einem Reset alle Flipflops diesen Wert enthalten.

Aufgabe 4.1 Einfach(st)e Ampel I – freie Zustandscodierung

Die zu entwerfende Ampel soll die vier Zustände A, B, C, D zyklisch durchlaufen.

a) Erstellen Sie einen Zustandsgraphen mit vier Zuständen 00, 01, 10, 11 für einen Automaten mit zwei Zustandsvariablen. Tragen Sie dabei gleich eine von Ihnen frei zu wählende Zustandscodierung und die Ausgangsvektoren in die Knoten des Graphen ein.

b) Leiten Sie von Ihrem Graphen eine Zustandsübergangstabelle ab, die zusätzlich die Ausgangsvektoren enthält.

c) Lesen Sie aus Ihrer Zustandsübergangstabelle die Schaltfunktionen ab und minimieren Sie diese gegebenenfalls mit Hilfe der Booleschen Algebra.

d) Realisieren Sie den Automaten mit HADES.

Aufgabe 4.2 Einfache Ampel II – direkte Zustandscodierung

Die zu entwerfende Ampel soll die vier Zustände A, B, C, D zyklisch durchlaufen. Als erster Zustand soll jedoch einmal der Zustand Z durchlaufen werden, also $Z, A, B, C, D, A, B, C, D, \ldots$

a) Erstellen Sie einen Zustandsgraphen mit fünf Zuständen für einen Automaten mit drei Zustandsvariablen. Definieren Sie die Zustandscodierung dabei derart, daß die Zustandsvariablen direkt zur Ansteuerung der Lampen verwendet werden können.

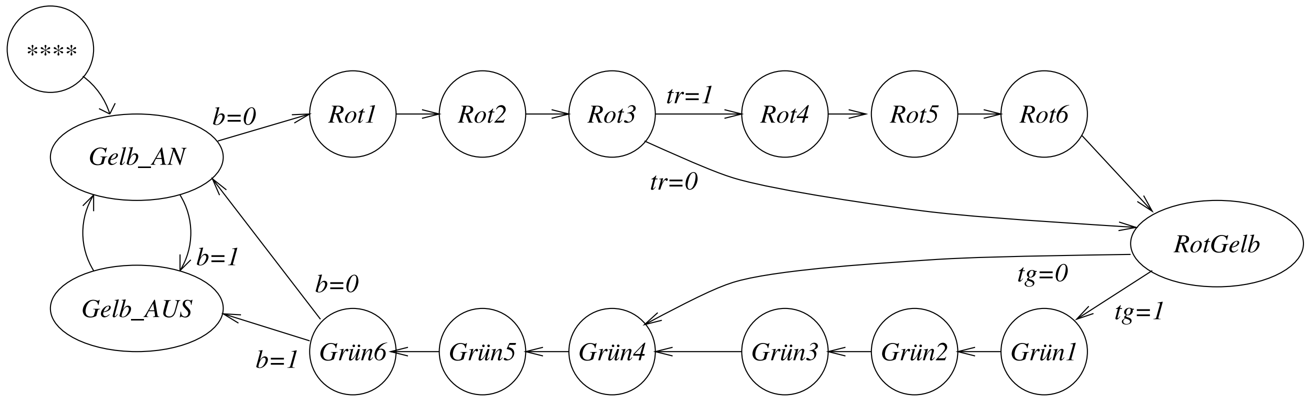
b) Leiten Sie von Ihrem Graphen die Zustandsübergangstabelle ab.

c) Lesen Sie aus Ihrer Zustandsübergangstabelle die benötigten Schaltfunktionen ab und minimieren diese mit Hilfe von KV-Diagrammen.

d) Realisieren und testen Sie den Automaten. Der Zustand `Z` sollte unmittelbar nach einem Reset der Flipflops angenommen werden.

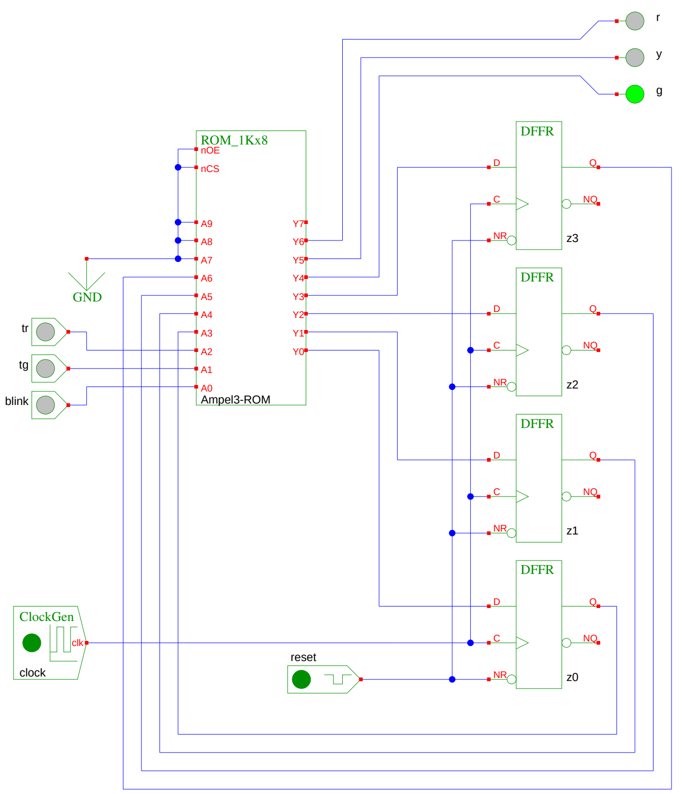
Aufgabe 4.3 Ampel mit steuerbaren Ampelphasen (optional)

Bei den beiden vorangegangenen Aufgaben wurden vier Zustände zyklisch durchlaufen, um die gewünschte Signalfolge zu erhalten. Bei Verkehrsampeln sind jedoch Gelb- und Rot-Gelb-Phase deutlich kürzer als Rot- und Grün-Phase. Dieser Aspekt wird nun berücksichtigt. Außerdem soll die Dauer (t_r) der Rot-Phase und die Dauer (t_g) der Grün-Phase durch je einen Eingang beeinflußbar sein, d.h. kurz oder lang. Ein dritter Eingang b steuert das Blinken der Ampel.



a) Definieren Sie eine Zustandscodierung für den dargestellten Graphen und übertragen Sie damit die Kanten des Graphen in die Zeilen der folgenden Tabelle:

Z	z_3	z_2	z_1	z_0	t_r	t_g	b	Rot	$Gelb$	$Grün$	z_3^+	z_2^+	z_1^+	z_0^+	Z^+
$Gelb_{AN}$ $Gelb_{AN}$															$Rot1$ $Gelb_{AUS}$
$Gelb_{AUS}$															$Gelb_{AN}$
$Rot1$															$Rot2$
$Rot2$															$Rot3$
$Rot3$ $Rot3$															$RotGelb$ $Rot4$
$Rot4$															$Rot5$
$Rot5$															$Rot6$
$Rot6$															$RotGelb$
$RotGelb$ $RotGelb$															$Grün4$ $Grün1$
$Grün1$															$Grün2$
$Grün2$															$Grün3$
$Grün3$															$Grün4$
$Grün4$															$Grün5$
$Grün5$															$Grün6$
$Grün6$ $Grün6$															$Gelb_{AN}$ $Gelb_{AUS}$
$****$															$Gelb_{AN}$
	A_6	A_5	A_4	A_3	A_2	A_1	A_0	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
	$ROM-Adresse$							$ROM-Daten$							

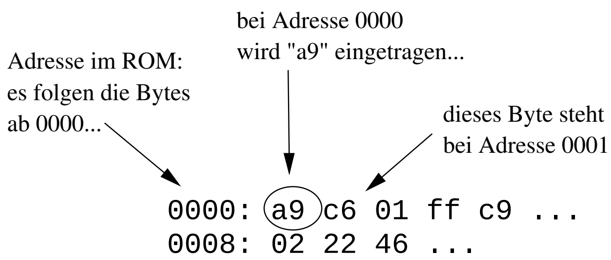


Der Automat soll mit einem Mikroprogramm-ROM realisiert werden. Jede Zeile in der Tabelle repräsentiert eine Kante des Graphen, d.h. einen Zustandsübergang. Eingangsvariablen, die nicht an Kanten des Graphen notiert sind, können 0 oder 1 sein – beide Fälle müssen jedoch im Mikroprogramm-ROM berücksichtigt werden. So ergeben sich insgesamt $2^7 = 128$ Einträge im ROM (von Adresse `0000` bis Adresse `007F`).

b) Lesen Sie aus Ihrer Zustandsübergangstabelle das Mikroprogramm ab und notieren Sie das Mikroprogramm in hexadezimaler Schreibweise in folgender Form:

```
AAAA : DD dd dd dd dd dd dd dd
aaaa : dd dd dd dd dd dd dd dd
              .
              .
```

Dabei stellt `AAAA` jeweils die Adresse des ersten Bytes `DD` dar. Die nachfolgenden Bytes stehen in den Adressen `AAAA+1, AAAA+2, ...` `A` bzw. `a`, stehen für eine vier Adressbits umfassende Hexadezimalstelle und `D` bzw. `d` für vier Datenbits. Ein Beispiel:



Am einfachsten kann so ein "Hex-Dump" mit einem ASCII-Editor erstellt werden (`axe`, `emacs`, `vi`, ...). Damit HADES ein derartiges "Hex-Dump" einlesen kann, müssen alle Leerzeichen entfallen. Anschließend kann die Datei in das ROM eingelesen werden, indem mit der Taste `E` der Konfigurationsdialog für den Baustein aufgerufen und der Dateiname[1] angegeben wird.

c) Realisieren Sie die mikroprogrammgesteuerte Ampel und simulieren Sie den Automaten.

[1] Die Angabe des Dateinamens entspricht derzeit nicht den üblichen Konventionen; sie wird relativ zum HADES-Verzeichnis interpretiert. Ein Beispiel: `/hades/ampel.rom` ist richtig, auch wenn die Datei tatsächlich in `/home/j95/5bitterl/hades/ampel.rom` liegt.